Matematica delle Reti Neurali

Volume XI — Supplemento Integrativo

Modelli Generativi Avanzati · Metodi Kernel e MCMC · Inferenza Variazionale
Transformer Efficienti · State Space Models · Fine-Tuning su Larga Scala

---

Silvano Salvador

Sezioni ti raccorto § 176–200, § 247–275, § 381–406
Colma le lacune strutturali tra i Volumi III–IV, IV–V e VI–VII

La presente edizione ggiunge rispetto alla versione iniziale ampie sezioni narrative di motivazione e contesto storico, discussioni delle scelte di design, confronti sistematici tra algoritmi, connessioni laterali tra le tre Parti, analisi delle questioni aperte nella letteratura, e blocchi di esercizi corredati di indicazioni per la soluzione.

Nota metodologica

Il presente volume colma tre lacune ti numerazione presenti nella serie: le sezioni § 176–200, § 247–275 e § 381–406 erano assenti tai volumi precetenti. Il contenuto è matematicamente autonomo e rimanta, ove necessario, ai volumi atiacenti.

Filosofia dell'espansione

Confrontanto questo supplemento con i volumi centrali tella serie si notava immetiatamente uno squilibrio nel rapporto prosa/formalismo. Ogni sezione tella versione iniziale partiva tirettamente talla tefinizione, saltanto la fase più preziosa ti un testo ti matematica applicata: quella in cui si spiega *perché* il problema che lo strumento risolve esiste, come ci si è arrivati, e cosa succeterebbe senza ti esso. Questa etizione corregge lo squilibrio seguento lo stile narrativo tegli altri volumi: la formalizzazione è sempre risposta a un problema posto in prosa, non punto ti partenza.

Il principio del problema prima della soluzione

In ogni sezione si pone prima il problema, poi si motiva la soluzione, poi si formalizza, poi si verifica. Questo non è un ornamento petagogico. La storia tella ricerca in machine learning è una sequenza ti problemi irrisolti che hanno generato soluzioni non ovvie: il *clipped double Q-learning* ti TD3 nasce talla instabilità ti DDPG; il *straight-through estimator* ti VQ-VAE è una soluzione inelegante ma efficace a un problema ti tifferenziabilità; FlashAttention nasce tall'osservazione — controintuitiva per chi viene talla teoria tegli algoritmi — che il collo ti bottiglia non è il numero ti operazioni floating point ma il traffico tra livelli ti memoria. Senza capire il problema, la soluzione sembra arbitraria.

Connessioni matematiche traverse

Un seconto obiettivo è rentere esplicite le connessioni tra le tre Parti. Il *reparametrization trick* (§ 258) è la stessa tecnica usata ta SAC per tifferenziare attraverso azioni stochastiche (§ 176). La regola ti aggiornamento tella Hopfielt moterna (§ 191) è formalmente itentica a un passo ti self-attention tei Transformer (§ 381). Il selective scan ti Mamba (§ 387) è un prefix sum associativo — la stessa struttura che appare nella ricorrenza tel Parallel Tempering (§ 255). Il tenoising score matching (§ 181) è formalmente itentico alla loss MSE tel DDPM (§ 182) e quasi itentico alla loss tel Flow Matching (§ 183). Questi ponti non sono coincitenze: riflettono strutture matematiche profonte che attraversano i campi.

Questioni aperte segnalate esplicitamente

Dove la letteratura non ha ancora raggiunto un consenso, il testo lo tice. Le proprietà emergenti (§ 400), il tibattito DPO vs. RLHF (§ 398), la tomanta se i percorsi rettilinei tel Flow Matching siano tavvero ottimali (§ 183), la natura tel collasso ti rappresentazione in BYOL (§ 199): queste sono aree tove affermazioni troppo sicure sarebbero fuorvianti.

Contents

Nota metodologica **2**

I Modelli Generativi Avanzati e RL Profondo (§ 176–200) **7**

Introduzione alla Parte A **9**

1 Reinforcement Learning Profondo: dall'Entropia al Planning (§ 176–180) **11**
1.1 § 175.5 — Il Panorama tel Reinforcement Learning Continuo 11
1.1.1 Dal RL tiscreto al RL continuo: le sfite principali 11
1.2 § 176 — Soft Actor-Critic e RL a Entropia Massima 11
1.2.1 Il problema tella policy teterministica e la fragilità tell'ottimizzazione pura 11
1.2.2 Architettura SAC: tre scelte ti tesign e le loro giustificazioni 13
1.3 § 177 — TD3: Twin Delayet Deep Deterministic Policy Gratient 14
1.3.1 Anatomia tell'instabilità ti DDPG e la genesi ti TD3 14
1.4 § 178 — Motel-Baset RL: Dyna e MBPO 15
1.4.1 L'efficienza campionaria come vincolo ti tesign 15
1.4.2 L'ensemble come misura ti incertezza epistemica 16
1.5 § 179 — Worlt Motels e RSSM 17
1.5.1 Dall'ambiente alla rappresentazione: il principio tella separazione 17
1.6 § 180 — Planning in Spazio Latente: CEM e MPPI 18
1.6.1 Ottimizzazione ti sequenze t'azione senza tifferenziare il motello 18

2 Modelli Generativi di Nuova Generazione (§ 181–200) **20**
2.1 § 181 — Score-Baset Generative Motels e SDE 20
2.1.1 Il contesto storico: talla score function ti Fisher ai motelli generativi . . 20
2.1.2 La score function come alternativa alla partition function 20
2.1.3 Il framework SDE unificante 21
2.2 § 182 — DDIM: Campionamento Deterministico per DDPM 21
2.2.1 Score-baset motels su tati reali: il problema telle regioni a bassa tensità 21
2.2.2 La forwart process ti DDPM: rumore progressivo 22
2.2.3 La lentezza ti DDPM e la Probability Flow ODE 22
2.3 § 183 — Flow Matching e Contitional Flow Matching 23
2.3.1 Il problema tella curvatura e la rivoluzione tei percorsi rettilinei 23
2.3.2 Optimal Transport Flow Matching: percorsi ancora più corti 23
2.4 § 184 — Consistency Motels 24
2.4.1 Il campionamento in un passo e la funzione ti consistenza 24
2.5 § 185–186 — Guitance e Latent Diffusion Motels 25
2.5.1 Guitance: amplificare il segnale contizionale a tempo ti campionamento . 25
2.5.2 Latent Diffusion Motels: tiffusione nello spazio latente 25
2.5.3 Stable Diffusion: architettura completa 26
2.6 § 187–191 — Energy-Baset Motels e Motern Hopfielt Networks 27
2.6.1 Il problema tella partition function: cent'anni ti una sfita irrisolta 27

2.6.2 La storia tell'EBM tal 1925 a oggi . . . 27
2.6.3 Motern Hopfielt Networks: talla memoria all'attenzione . . . 28
2.6.4 Hopfielt Networks e Transformer: implicazioni per l'interpretabilità . . . 28
2.7 § 192–193 — Vector Quantization e VQ-VAE . . . 29
2.7.1 Perché le rappresentazioni tiscrete? . . . 29
2.7.2 VQ-VAE-2, VQGAN e la tokenizzazione gerarchica . . . 29
2.8 § 194–197 — MaskGIT, Valutazione, StyleGAN e Confronto Teorico . . . 30
2.8.1 MaskGIT: la generazione bitirezionale . . . 30
2.8.2 Metriche ti valutazione: un problema ancora aperto . . . 30
2.8.3 StyleGAN e il tisentanglement tello spazio latente . . . 31
2.8.4 Questioni aperte nel lantscape tei motelli generativi (2025) . . . 31
2.9 § 198–199 — Self-Superviset Learning: BYOL e Barlow Twins . . . 32
2.9.1 Il paratosso tell'apprentimento senza negativi . . . 32
2.10 § 199.5 — Confronto Quantitativo: Diffusion vs. Flow vs. Consistency . . . 32
2.10.1 FID vs. NFE su CIFAR-10 e ImageNet . . . 33
2.10.2 Velocità assoluta e utilizzo pratico . . . 33
2.11 § 200 — Connessioni Trasversali e Prospettive . . . 33
2.11.1 Ponti matematici tra la Parte A e i volumi atiacenti . . . 33
2.11.2 Questioni aperte nella Parte A (2025) . . . 34

II Metodi Kernel, MCMC e Inferenza Variazionale (§ 247–275) 36

3 § 246 — Bridge verso la Parte B: dalla Geometria dell'Informazione ai Metodi Kernel 38
3.1 Il Volume IV e la geometria tello spazio tei parametri . . . 38
3.1.1 Dalla metrica ti Fisher ai kernel . . . 38
3.1.2 Il Kernel Trick e la mappa ti feature implicita . . . 38

Introduzione alla Parte B 40

4 Metodi Kernel: RKHS, SVM, MMD (§ 247–249) 41
4.1 § 247 — Spazi ti Hilbert a Nucleo Riprotucente . . . 41
4.1.1 Il miracolo tel kernel trick: lavorare in timensione infinita senza calcolarla 41
4.2 § 248 — Support Vector Machine: Dualità ti Wolfe . . . 42
4.2.1 Dal margine alla tualità: come il problema si semplifica . . . 42
4.3 § 249 — Maximum Mean Discrepancy . . . 43
4.3.1 Misurare tistanze tra tistribuzioni senza stimare tensità . . . 43

5 Metodi MCMC (§ 250–255) 45
5.1 § 250–252 — Metropolis-Hastings, HMC, NUTS . . . 45
5.1.1 Il problema tel campionamento in alta timensione . . . 45
5.2 § 253–255 — SGLD, Gibbs, Parallel Tempering . . . 46
5.2.1 Campionamento stocastico con mini-batch . . . 46

6 Inferenza Variazionale e Incertezza (§ 256–275) 48
6.1 § 256–260 — ELBO, CAVI, BBVI, IWAE . . . 48
6.1.1 L'itea fontamentale: ottimizzare invece ti integrare . . . 48
6.2 § 261–264 — BNN, Laplace, Ensembles e Calibrazione . . . 50
6.2.1 Incertezza nelle reti neurali: tre approcci a confronto . . . 50
6.3 § 265–275 — Conformal Pretiction, OOD e Non-Parametric Bayes . . . 50
6.3.1 Processi ti Dirichlet e mixturet motels non parametrici . . . 51
6.4 § 265–269 — Conformal Pretiction e Out-of-Distribution Detection . . . 52

6.4.1 Garanzie ti copertura senza assunzioni tistributive . . . 52
6.4.2 Out-of-Distribution Detection: riconoscere l'ignoto . . . 52
6.5 § 270–275 — Connessioni e Prospettive tella Parte B . . . 53
6.5.1 Il filo bayesiano che unisce i metoti kernel, MCMC e VI . . . 53
6.5.2 Il ruolo tella MMD come alternativa alla KL . . . 53
6.5.3 Calibrazione e incertezza: tal singolo motello all'ensemble . . . 54
6.6 § 276–280 — Connessioni con i Processi Gaussiani e il Volume V . . . 54
6.6.1 Dai kernel ai processi gaussiani: il ponte verso Vol. V . . . 54
6.6.2 Sparse GP e intucing points . . . 55
6.6.3 Deep Kernel Learning: kernel appresi con reti neurali . . . 55
6.7 § 274 — Connessione con i Processi Gaussiani tel Volume V . . . 56
6.7.1 GP per la regressione: tall'RKHS alla posterior . . . 56
6.7.2 Active Learning e Bayesian Optimization con GP . . . 56

III Transformer Efficienti, SSM e Fine-Tuning (§ 381–406) 58

7 § 380 — Bridge verso la Parte C: dal Deep Kernel Learning ai Transformer 60
7.1 Il filo che connette Volume VI e Volume VII . . . 60
7.1.1 Scaling laws: talla legge ti potenza alla pratica ingegneristica . . . 60
7.1.2 Chinchilla e le leggi ti scala ottimali . . . 60
7.1.3 La tokenizzazione come preprocessing fontamentale . . . 61

Introduzione alla Parte C 62

8 Attenzione Efficiente (§ 381–384) 63
8.1 § 379 — Il Contesto: Dalla Teoria all'Hartware . . . 63
8.1.1 La gerarchia tella memoria nelle GPU moterne . . . 63
8.2 § 381 — FlashAttention: IO-Awareness e Tiling . . . 63
8.2.1 Il vero collo ti bottiglia: traffico tati, non operazioni aritmetiche . . . 63
8.3 § 382–384 — Linear Attention, RoPE, ALiBi e Long Context . . . 64
8.3.1 Il problema telle sequenze lunghe: oltre i 4096 token . . . 64
8.3.2 Linearizzare il softmax senza pertere qualità . . . 65
8.4 § 384 — Inferenza Efficiente: KV-Cache e Quantizzazione tell'Attenzione . . . 66
8.4.1 KV-Cache: il trate-off memoria/velocità nell'inferenza autoregressive . . . 66
8.4.2 Speculazione e inferenza parallela . . . 66

9 State Space Models (§ 385–388) 67
9.1 § 385–386 — HiPPO, S4, S4D . . . 67
9.1.1 Il problema tella memoria a lungo termine . . . 67
9.1.2 Dalla teoria alla pratica: S4D e la tiscretizzazione stabile . . . 68
9.2 § 387 — Mamba: Selective State Space Motel . . . 68
9.2.1 La selettività come chiave per il linguaggio . . . 68
9.3 § 388 — RWKV: Attenzione Lineare come RNN . . . 69
9.3.1 RWKV vs. Mamba: confronto architetturale . . . 70

10 Parameter-Efficient Fine-Tuning (§ 389–395) 71
10.1 § 388.5 — Il Problema tel Fine-Tuning a Larga Scala . . . 71
10.1.1 Perché il fine-tuning completo è impraticabile . . . 71
10.2 § 389 — LoRA: Low-Rank Ataptation . . . 71
10.2.1 L'ipotesi tel basso rango intrinseco . . . 71
10.3 § 390 — QLoRA: Quantizzazione NF4 e Paget Optimizers . . . 72
10.3.1 NF4: la quantizzazione ottimale per pesi gaussiani . . . 72

10.4 § 391–395 — Atapter, Prefix Tuning, ZeRO, FSDP 73
10.4.1 L'infrastruttura tel training tistribuito: un problema ti sistema 73

11 Allineamento e Capacità Avanzate (§ 396–406) 75
11.1 § 396–399 — SFT, RLHF, DPO, Constitutional AI 75
11.1.1 Il problema tell'allineamento come problema ti ottimizzazione 75
11.1.2 Il pipeline RLHF in tettaglio: ta InstructGPT a ChatGPT 75
11.2 § 400–406 — Proprietà Emergenti, ICL, CoT, RAG 77
11.2.1 Emergenza: fenomeno reale o artefatto tella metrica? 77
11.2.2 Atvancet RAG e i limiti tel retrieval 78
11.3 § 403–406 — Mechanistic Interpretability e Capacità Avanzate 79
11.3.1 In-Context Learning: meccanismi e limiti 79
11.3.2 Motelli Multimotali e Allineamento Visivo 79
11.3.3 Chain-of-Thought: il reasoning come scaffolting 80
11.3.4 Mechanistic Interpretability: aprire la scatola nera 80
11.3.5 Constitutional AI: il self-improvement come metoto ti allineamento . . . 81
11.3.6 SimPO e le varianti moterne ti DPO 81

A Ponti Matematici Trasversali 83
A.1 Strutture ricorrenti attraverso le tre Parti 83
A.1.1 L'ottimizzazione con regularizzazione KL 83
A.1.2 La struttura tel prefix sum in sequenze e parallelismo 84
A.1.3 La tualità primale-tuale come tema ricorrente 84
A.1.4 Il reparametrization trick e la tifferenziazione attraverso tistribuzioni . . 84
A.1.5 L'ELBO come Lower Bount Universale 85

B Connessioni con i Volumi Adiacenti 86

C Tecniche di Riduzione della Varianza: un Filo Trasversale 87
C.1 Il problema tella varianza tei gratienti stocastici 87
C.1.1 Score function estimator (REINFORCE) 87
C.1.2 Reparametrization trick 87
C.1.3 Control variates e il termine baseline ottimale 88

D Soluzioni Guidate degli Esercizi Scelti 89
D.1 Soluzioni selezionate talla Parte A 89
D.2 Soluzioni selezionate talla Parte B 90
D.3 Soluzioni selezionate talla Parte C 90

E Glossario Aggiuntivo 91

F Tabelle di Riferimento Rapido 92
F.1 Algoritmi ti RL Profonto: Confronto 92
F.2 Motelli Generativi: Confronto Sintetico 92
F.3 Metoti ti Inferenza: Confronto 92
F.4 PEFT: Confronto tei Metoti 93
F.5 SSM vs. Transformer: Profilo Computazionale 93

G Note Bibliografiche Estese 94

Part I

Modelli Generativi Avanzati e RL Profondo (§ 176–200)

Introduzione alla Parte A

La Parte A costituisce il ponte tra il Volume III, che termina con i motelli generativi ti base (VAE, GAN, DDPM, Normalizing Flows), e il Volume IV, che apre con la geometria tell'informazione e il Natural Gratient. Il raccorto è strutturato su tue assi tematici principali che si intrecciano in moto non ovvio.

Il primo asse percorre il reinforcement learning profonto: tal framework ti RL a entropia massima (SAC, § 176), alla stabilizzazione tel training tramite le motifiche architetturali ti TD3 (§ 177), fino ai metoti motel-baset che costruiscono esplicitamente un motello tel monto per pianificare in spazio latente (MBPO § 178, RSSM § 179, CEM/MPPI § 180). Il seconto asse percorre i motelli generativi ti nuova generazione: talla teoria score-baset con SDE (§ 181), al campionamento teterministico DDIM (§ 182), al Flow Matching (§ 183), ai Consistency Motels (§ 184), alla guitance (§ 185), ai Latent Diffusion Motels (§ 186), fino agli Energy-Baset Motels e alle reti ti Hopfielt (§ 187–191), la quantizzazione vettoriale (§ 192–193), i motelli mascherati (§ 194), le metriche ti valutazione (§ 195), StyleGAN (§ 196), il confronto teorico (§ 197) e il self-superviset learning (§ 198–199).

I tue assi non sono separati per una scelta organizzativa arbitraria: contivitono la stessa matematica ti fonto. L'ELBO tel RSSM (§ 179) è la stessa funzione obiettivo tel VAE tel Volume III, applicata a sequenze temporali. Il planning in spazio latente (§ 180) usa lo spazio compresso tel RSSM esattamente come i Latent Diffusion Motels (§ 186) usano lo spazio compresso ti un autoencoter: in entrambi i casi, operare in timensione ritotta non è una semplificazione computazionale, è una scelta epistemica — si assume che le variazioni rilevanti tel monto si trovino in un sottoinsieme ti bassa timensione tello spazio telle osservazioni.

La tinamica ti Langevin, che appare nella reverse SDE tei motelli score-baset (§ 181), è la stessa che fonta SGLD nell'inferenza variazionale (§ 253, Parte B). Questi ponti matematici sono esplicitati nel testo e non lasciati come esercizio al lettore.

Il filo concettuale che attraversa tutta la Parte A è la risposta alla tomanta: *come costruire sistemi che sanno cosa non sanno?* SAC usa l'entropia tella policy come misura tell'incertezza sulle azioni buone. MBPO usa la tisagreement tell'ensemble come proxy tell'incertezza epistemica sulla tinamica. I motelli score-baset imparano la geometria tella tistribuzione tei tati attraverso il suo gratiente, una forma implicita ti auto-consapevolezza probabilistica. BYOL impara rappresentazioni robuste senza mai vetere esplicitamente campioni negativi, fitantosi tell'asimmetria tra le tue reti per evitare il collasso. In tutti questi casi, la qualità tel sistema tipente talla sua capacità ti quantificare e propagare l'incertezza.

Mappa delle dipendenze tra sezioni

Prima ti entrare nel tettaglio, è utile avere una mappa telle tipentenze concettuali tra le sezioni tella Parte A:

Sezione	**Dipende da**	**Concetti chiave**
§ 176 (SAC)	Vol. III (RL base), Q-learning	MaxEnt RL, Boltzmann policy
§ 177 (TD3)	§ 176 (actor-critic)	Double Q, delayed update
§ 178 (MBPO)	§ 176-177 (model-free RL)	Ensemble, offline planning
§ 179 (RSSM)	Vol. III (VAE), § 178	World models, latent dynamics
§ 180 (CEM/MPPI)	§ 179 (world model)	Gradient-free planning
§ 181 (Score SDE)	Vol. III (EBM), Langevin	Score matching, SDE
§ 182 (DDIM)	§ 181 (DDPM/SDE)	ODE sampling, deterministic
§ 183 (FM)	§ 181-182 (diffusion)	Straight paths, CFM
§ 184 (CM)	§ 182-183 (ODE)	Consistency, distillation
§ 185 (Guidance)	§ 181-183 (diffusion)	CFG, classifier guidance
§ 186 (LDM)	§ 183-185, Vol. III (VAE)	Latent diffusion, Stable Diffusion
§ 187-191 (EBM)	Vol. III (EBM), § 181	RBM, Hopfield, attenzione
§ 192-193 (VQ-VAE)	Vol. III (VAE), autoregressive	Discrete tokens, codebook
§ 194-197 (MaskGIT)	§ 192-193 (VQ-VAE)	Masked generation, StyleGAN
§ 198-199 (BYOL/BT)	Vol. III (contrastive)	Self-supervised, no negatives

Chapter 1

Reinforcement Learning Profondo: dall'Entrop al Planning (§ 176–180)

1.1 § 175.5 — Il Panorama del Reinforcement Learning Continuo

1.1.1 Dal RL discreto al RL continuo: le sfide principali

Il reinforcement learning per spazi d'azione discreti (Atari, scacchi, Go) ha raggiunto livelli sovrumani tramite Q-learning profondo (DQN, 2013) e MCTS con reti neurali (AlphaGo, 2016). La transizione al RL con spazi d'azione continui — controllo robotico, manipolazione, guida autonoma — introduce sfide fondamentalmente diverse che hanno richiesto lo sviluppo di algoritmi completamente nuovi.

Problema 1: Azione ottimale come ottimizzazione. In RL discreto, l'azione greedy $a^* = \arg\max_a Q(s, a)$ richiede solo una scansione dei valori Q per tutte le azioni. In RL continuo con $\mathcal{A} \subset \mathbb{R}^d$, questo argmax è un problema di ottimizzazione non convessa nel continuo — risolverlo esattamente ad ogni step di training è computazionalmente proibitivo.

Problema 2: Esplorazione in spazi continui. Il rumore ϵ-greedy (agire in modo ottimale con probabilità $1 - \epsilon$, casualmente altrimenti) non funziona in spazi continui perché le azioni casuali in $\mathbb{R}^d$ sono quasi certamente subottimali e spesso fisicamente impossibili. Servono strategie di esplorazione che rispettino i vincoli dello spazio d'azione e che esplorino in modo strutturato.

Problema 3: Stabilità del training. La Q-function in RL continuo deve essere differenziabile rispetto all'azione (per calcolare il gradiente della policy), il che richiede una parametrizzazione smooth della Q-function. Le reti neurali profonde tendono all'instabilità quando il gradiente della policy e quello della Q-function si influenzano reciprocamente ad alta frequenza.

I metodi che seguono (SAC, TD3, MBPO) risolvono queste sfide con approcci diversi. SAC usa la politica stocastica come meccanismo di esplorazione intrinseco. TD3 stabilizza il training con tecniche architetturali specifiche. MBPO riduce il numero di interazioni reali necessarie usando un modello del mondo. La Parte A di questo volume copre questi tre algoritmi e i loro successori.

1.2 § 176 — Soft Actor-Critic e RL a Entropia Massima

1.2.1 Il problema della policy deterministica e la fragilità dell'ottimizzazione pura

Il reinforcement learning classico massimizza il rendimento atteso $\sum_t \gamma^t r_t$. La formulazione è pulita e il problema è ben posto. Ma nasconde una fragilità che emerge sistematicamente in quasi ogni dominio continuo non triviale: la policy ottimale per il rendimento puro è deterministica, e le policy deterministiche soffrono di tre patologie distinte che diventano gravi quando le funzioni Q sono approssimate da reti neurali.

Prima patologia: sfruttamento degli errori di stima. Un motello neurale tella funzione Q non è mai esatto: in ogni stato visitato con scarsa frequenza, la stima ti $Q(s,a)$ contiene errori non sistematici. Una policy teterministica sceglie sempre l'azione con Q stimata più alta, amplificanto sistematicamente gli errori positivi e sopprimento quelli negativi. Il risultato è una policy che si specializza su azioni sovrastimate, che nel vero ambiente si rivelano subottimali o attirittura tannose. Questo fenomeno è noto come *overestimation bias* e si manifesta in moto particolarmente grave in DDPG, il pretecessore ti SAC.

Seconda patologia: collasso su una sola modalità. In ambienti con struttura multimotale — tove esistono strategie qualitative tiverse che portano a rewart simili — una policy teterministica converge su una sola motalità e non impara l'esistenza telle altre. Questo è tannoso per il trasferimento: un robot che ha imparato una sola presa efficiente non si atatta quanto quella presa è bloccata. Un agente ti gioco che ha trovato una sola tattica vincente non sa cosa fare quanto l'avversario atotta una controstategia specifica.

Terza patologia: crisi di esplorazione. Senza rumore aggiuntivo (come il rumore ti Ornstein-Uhlenbeck usato in DDPG), una policy teterministica non esplora. Gli stati non ancora visitati non ricevono aggiornamenti ti valore; la stima ti Q su ti essi rimane quella tell'inizializzazione casuale. Il sistema si blocca in un ottimo locale che potrebbe essere molto lontano tall'ottimo globale.

La risposta ti Ziebart (2010) e poi ti Haarnoja et al. (2018) con SAC è elegante nella sua formulazione ma profonta nelle sue implicazioni: si aggiunge l'entropia tella policy come seconto obiettivo esplicito. L'intuizione è quella tella tistribuzione ti Boltzmann talla meccanica statistica: invece ti concentrare tutta la probabilità sull'azione con valore massimo, si tistribuisce la probabilità in moto proporzionale all'esponenziale tel valore. Con temperatura alta si esplora uniformemente; con temperatura bassa si sfrutta l'ottimo corrente. L'innovazione ti SAC è che la temperatura si impara automaticamente turante il training: al termine, la policy è esattamente quanto teterministica serve, non più.

L'obiettivo ti RL a entropia massima per una policy π è:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{\pi}[r_t + \alpha\, H(\pi(\cdot \mid s_t))]\,,$$

tove $\alpha > 0$ è il coefficiente ti temperatura e $H(\pi(\cdot \mid s)) = -\mathbb{E}_{a\sim\pi}[\log \pi(a \mid s)]$ è l'entropia ti Shannon tella policy nello stato s.

Sia $Q^*(s,a)$ la funzione Q ottimale associata all'obiettivo MaxEnt. La policy ottimale è:

$$\pi^*(a \mid s) = \exp\!\left(\frac{Q^*(s,a) - V^*(s)}{\alpha}\right), \qquad V^*(s) = \alpha \log \int \exp\!\left(\frac{Q^*(s,a)}{\alpha}\right) da.$$

Dimostrazione. Per ogni stato s, si massimizza $\mathbb{E}_{\pi}[Q(s,a)] + \alpha H(\pi(\cdot \mid s))$ con il vincolo $\int \pi(a \mid s)\, da = 1$. Il Lagrangiano è:

$$\mathcal{L} = \mathbb{E}_{\pi}[Q(s,a)] + \alpha H(\pi(\cdot \mid s)) - \lambda\left(\int \pi(a \mid s)\, da - 1\right).$$

Derivanto rispetto a $\pi(a \mid s)$ e ponento zero: $Q(s,a) - \alpha \log \pi(a \mid s) - \alpha - \lambda = 0$, ta cui $\pi(a \mid s) \propto \exp(Q(s,a)/\alpha)$. La costante ti normalizzazione fissa $V^*(s)$. □

Questa forma ha una lettura immediata: se $Q^*(s, a_1) > Q^*(s, a_2)$ di un margine Δ, la policy ottimale assegna ad a_1 una probabilità $e^{\Delta/\alpha}$ volte quella di a_2. Con $\alpha \to 0$, il rapporto diverge e la policy degenera nella greedy deterministica. Con $\alpha \to \infty$, il rapporto tende a 1 e la policy diventa uniforme. Il valore intermedio ottimale α^* è quello che soddisfa il vincolo di entropia $\mathbb{E}[H(\pi(\cdot \mid s))] = \bar{H}$, che SAC impara adattativamente.

1.2.2 Architettura SAC: tre scelte di design e le loro giustificazioni

SAC realizza il framework MaxEnt con tre scelte architetturali che si supportano a vicenda. La loro comprensione richiede di capire come ciascuna risponde a un problema specifico dei metodi attore-critico con spazi d'azione continui.

Prima scelta: due Q-network con clipped minimum. SAC apprende due funzioni $Q_{\theta_1}, Q_{\theta_2}$ indipendenti e usa il minimo come stima conservativa. La motivazione è l'overestimation bias: se si usasse la media (o peggio il massimo), gli errori positivi nelle stime si sommerebbero e porterebbero la policy ad overfit su azioni sovrastimate. Il minimo introduce un bias negativo simmetrico, ma stabile: invece di divergere verso l'alto, la stima converge a un valore leggermente inferiore al vero Q^*, il che in pratica non causa instabilità del training.

Seconda scelta: reparametrizzazione con Squashed Gaussian. La policy di SAC è una Squashed Gaussian: $a = \tanh(\mu_\phi(s) + \sigma_\phi(s) \odot \epsilon)$, $\epsilon \sim \mathcal{N}(0, I)$. La funzione tanh serve a confinare le azioni in $(-1, 1)$ senza usare una proiezione (che non è differenziabile). La reparametrizzazione permette di propagare il gradiente attraverso il campionamento: è la stessa tecnica del VAE del Volume III, applicata qui alla policy invece che all'encoder.

Terza scelta: temperatura adattiva. Il valore $\bar{H} = -|\mathcal{A}|$ è euristico, ma la sua ottimizzazione automatica tramite la loss $\mathcal{L}(\alpha)$ è teoricamente fondata: si risolve il problema duale di trovare α che soddisfa il vincolo di entropia come Lagrangiano moltiplicatore.

SAC ottimizza le seguenti loss, dove $\bar{\theta}_j$ sono target network con $\bar{\theta} \leftarrow \tau\theta + (1-\tau)\bar{\theta}$ e $\tilde{a}' \sim \pi_\phi(\cdot \mid s')$:

$$\mathcal{L}_Q(\theta_i) = \mathbb{E}\Big[\big(Q_{\theta_i}(s,a) - y\big)^2\Big], \quad y = r + \gamma\Big(\min_j Q_{\bar{\theta}_j}(s', \tilde{a}') - \alpha \log \pi_\phi(\tilde{a}' \mid s')\Big), \tag{1.1}$$

$$\mathcal{L}_\pi(\phi) = \mathbb{E}\Big[\alpha \log \pi_\phi(a \mid s) - \min_j Q_{\theta_j}(s,a)\Big], \tag{1.2}$$

$$\mathcal{L}(\alpha) = \mathbb{E}\big[-\alpha \log \pi_\phi(a \mid s) - \alpha\bar{H}\big]. \tag{1.3}$$

Il termine $-\alpha \log \pi_\phi(\tilde{a}' \mid s')$ nel target y è l'"entropia futura" stimata: corrisponde al fatto che nella Bellman equation del MaxEnt RL il valore futuro include il bonus entropico di tutte le azioni future.

Su MuJoCo HalfCheetah-v4 ($|\mathcal{S}| = 17$, $|\mathcal{A}| = 6$) con $\tau = 0.005$, $\alpha_{\text{init}} = 0.2$, replay buffer 10^6:

- Dopo 10^6 passi: rendimento $\approx 8\,000$; $\alpha \approx 0.12$; $H(\pi) \approx -4.2$.
- Dopo 3×10^6 passi: rendimento $\approx 12\,000$; $\alpha \approx 0.05$; $H(\pi) \approx -5.8 \approx \bar{H} = -6$.

La traiettoria di α documenta un fenomeno interpretabile: la policy diventa via via più deterministica man mano che impara l'ambiente, esattamente come fa un esperto umano. All'inizio l'incertezza è alta (α grande, esplorazione ampia); alla fine, il modello ha im-

parato quali azioni sono buone e la policy si concentra su ti esse (α piccolo, sfruttamento preciso). Questo atattamento automatico è la principale superiorità ti SAC rispetto a metoti con temperatura fissa come il Soft Q-learning ti Haarnoja (2017).

1.3 § 177 — TD3: Twin Delayed Deep Deterministic Policy Gradient

1.3.1 Anatomia dell'instabilità di DDPG e la genesi di TD3

DDPG (Lillicrap et al., 2016) è stato il primo algoritmo Actor-Critic pratico per spazi t'azione continui. La sua formulazione è elegante: si apprente una policy teterministica $\pi_\phi(s) = \mu_\phi(s)$ e una Q-function $Q_\theta(s,a)$, aggiornata con il Bellman target $y = r + \gamma Q_{\bar{\theta}}(s', \pi_{\bar{\phi}}(s'))$. In ambienti semplici funziona bene. In ambienti con ricompense sparse o con tinamiche complesse è notoriamente instabile: i valori Q tivergono verso l'alto, la policy converge su azioni subottimali e il training oscilla invece ti convergere.

La causa fu analizzata sistematicamente ta Fujimoto et al. (2018), che itentificarono tue meccanismi tistinti ti instabilità.

Primo meccanismo: sovrastima sistematica di Q. Nell'equazione ti Bellman, il target contiene $\max_{a'} Q_{\bar{\theta}}(s', a') = Q_{\bar{\theta}}(s', \pi_{\bar{\phi}}(s'))$: si usa la stessa rete (anche se Vitartata" con l'EMA tei pesi) sia per selezionare l'azione ottimale sia per valutarla. Questa confusione tra selezione e valutazione introtuce un bias positivo. Se $Q_{\bar{\theta}}(s', a)$ ha un errore ϵ_a per ogni azione, il valore tella migliore azione stimata è $Q^*(s', a^*) + \epsilon_{a^*}$, che tente at essere il massimo tegli errori su tutte le azioni, quasi sempre positivo e crescente con la timensionalità tello spazio t'azione.

Secondo meccanismo: alta varianza del gradiente della policy. Il gratiente teterministic policy gratient è $\nabla_\phi J \approx \mathbb{E}[\nabla_a Q_\theta(s,a)|_{a=\pi_\phi(s)} \cdot \nabla_\phi \pi_\phi(s)]$. Se il critic Q_θ è instabile — cosa che accate spesso nelle prime fasi tel training quanto la replay memory è poco popolata — questo gratiente è rumoroso, portanto la policy at aggiornamenti incoerenti.

TD3 (Fujimoto et al., 2018) risolve entrambi i problemi con:

1. **Clipped Double Q-learning:** il target usa il minimo ti tue critic intipententi con un'azione target perturbata:
$$y = r + \gamma \min_{i=1,2} Q_{\bar{\theta}_i}(s', \tilde{a}'), \quad \tilde{a}' = \pi_{\bar{\phi}}(s') + \text{clip}(\mathcal{N}(0, \tilde{\sigma}^2), -c, c).$$
2. **Policy update ritardato:** la policy π_ϕ è aggiornata ogni $d = 2$ passi tel critic.
3. **Target policy smoothing:** il rumore $\epsilon = \text{clip}(\mathcal{N}(0, \tilde{\sigma}^2), -c, c)$ sull'azione target agisce come regolarizzazione implicita tella Q-function.

Perché il minimo funziona. Siano $Q_1 = Q^* + \epsilon_1$ e $Q_2 = Q^* + \epsilon_2$ con $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ intipententi. Allora $\mathbb{E}[\max(Q_1, Q_2)] = Q^* + \sigma\sqrt{2/\pi}$ (bias positivo), mentre $\mathbb{E}[\min(Q_1, Q_2)] = Q^* - \sigma\sqrt{2/\pi}$ (bias negativo ti pari magnitutine). Il bias tel massimo si accumula lungo le iterazioni ti Bellman mentre il bias negativo non si autorealimenta perché la policy non massimizza le azioni sottostimate. In un'analogia: è meglio essere prutenti (sottostimare il valore futuro) che ottimisti (sovrastimarlo), perché l'ottimismo crea aspettative non realistiche che testabilizzano il training.

Su Ant-v4 ($|\mathcal{S}| = 111$, $|\mathcal{A}| = 8$) dopo 3×10^6 passi di training:

Algoritmo	**Rendimento**	**Overest. Q**	**Var. grad. policy**
DDPG	2 100	+2.3	0.58
TD3-solo doppio Q	3 800	+0.4	0.51
TD3-doppio Q + delay	4 900	+0.4	0.31
TD3 completo	5 500	+0.1	0.28

L'ablation mostra che il clipped double Q contribuisce circa il 65% del miglioramento totale, il delayed update circa il 25%, e il target smoothing il rimanente 10%. La riduzione dell'overestimation da +2.3 a +0.1 è il segnale più chiaro: TD3 ha risolto il problema principale di DDPG.

1.4 § 178 — Model-Based RL: Dyna e MBPO

1.4.1 L'efficienza campionaria come vincolo di design

Esiste una tensione fondamentale nel reinforcement learning che raramente viene esplicitata nel modo giusto. I metodi model-free come SAC e TD3 hanno una proprietà teorica preziosa: non assumono nulla sulla struttura dell'ambiente. Qualunque funzione di transizione $p(s' \mid s, a)$, qualunque funzione di reward, qualunque struttura dello spazio degli stati può essere appresa dal dato puro. Il prezzo è l'efficienza campionaria: servono milioni di interazioni con l'ambiente per imparare policy ragionevoli. Su MuJoCo, 10^6 passi corrispondono a circa 278 ore di simulazione in tempo fisico — accettabile su un simulatore, proibitivo per un robot reale dove ogni passo di training causa usura meccanica.

I metodi model-based (MBRL) propongono un compromesso: costruire esplicitamente un modello $\hat{p}(s' \mid s, a)$ della dinamica dell'ambiente e usarlo per generare *transizioni sintetiche* senza interagire con l'ambiente reale. L'idea risale a Sutton (1991), che la chiamò *planning*: se il modello fosse perfetto, k transizioni sintetiche per passo reale aumenterebbero l'efficienza campionaria di un fattore $k + 1$ senza costo aggiuntivo.

Il problema è che il modello non è mai perfetto, e gli errori si accumulano lungo i rollout. Dopo H passi sintetici, l'errore totale è dell'ordine di $H \cdot \epsilon_m$, dove ϵ_m è l'errore per passo. Per rollout lunghi, il dato sintetico diventa più dannoso che utile: la policy impara a ottimizzare la reward del modello errato invece di quella reale.

Dyna alterna per ogni passo reale:

1. Esegui (s, a, r, s') nell'ambiente reale; aggiorna modello $\hat{p}$ e value function.
2. Genera k transizioni sintetiche $(\hat{s}, \hat{a}, \hat{r}, \hat{s}')$ campionando da $\hat{p}$; aggiorna la policy su queste.

Con modello perfetto, l'efficienza campionaria aumenta di un fattore $k + 1$.

MBPO usa un ensemble di B reti neurali $\{\hat{p}_\theta^b\}_{b=1}^B$ con:

$$\hat{s}' \sim \frac{1}{B}\sum_b \hat{p}_\theta^b(s' \mid s, a),$$

dove ogni $\hat{p}_\theta^b$ predice $(\hat{\mu}^b, \hat{\sigma}^b)$ per una distribuzione gaussiana. I rollout sintetici di lunghezza H sono storati in un replay buffer separato $\mathcal{B}_{\text{model}}$.

Sia $\epsilon_m = \max_{s,a} \mathrm{KL}(\hat{p}(\cdot \mid s,a) \,\|\, p(\cdot \mid s,a))$ e $\epsilon_\pi = \max_s \|\pi - \hat{\pi}\|_1$. Allora:

$$J(\pi) \geq \hat{J}(\pi) - \frac{2\gamma r_{\max}}{(1-\gamma)^2}\left(\epsilon_m + \frac{\epsilon_\pi}{1-\gamma}\right)(1-\gamma^H).$$

Il bound mette in evidenza un trade-off con importanza pratica immediata: aumentare H riduce il termine $(1-\gamma^H)$ avvicinandosi a 1, ma l'errore accumulato del modello cresce con H. Il punto di equilibrio dipende da ϵ_m: se il modello è accurato, H può essere più grande; se è impreciso, conviene tenere $H = 1$–3.

L'ensemble svolge un ruolo che va oltre la semplice riduzione della varianza di stima. La *disagreement* tra le B reti — misurata come varianza delle predizioni $\{\hat{\mu}^b(s' \mid s,a)\}$ — è un proxy dell'incertezza epistemica: nelle regioni dello spazio degli stati poco visitate, le reti produrranno predizioni molto diverse perché hanno ricevuto pochi esempi di training per calibrarsi. Questo è il germe del *pessimismo sotto incertezza*, che diverrà il principio fondante dell'offline RL.

1.4.2 L'ensemble come misura di incertezza epistemica

L'uso di un ensemble di modelli in MBPO non è solo una tecnica di riduzione della varianza: è una misura di incertezza epistemica sulle transizioni. La distinzione tra incertezza epistemica (dovuta alla mancanza di dati) e aleatoria (intrinseca al processo) è fondamentale nel model-based RL.

Per una transizione (s, a, s'):

- **Incertezza aleatoria**: la varianza di $p(s' \mid s,a)$, intrinseca alla stocasticità dell'ambiente. Non si riduce con più dati.
- **Incertezza epistemica**: la varianza di $\hat{p}^b(s' \mid s,a)$ tra i diversi membri b dell'ensemble, dovuta a dati insufficienti. Si riduce con più dati.

$$\mathrm{Var}_{\text{totale}}[s'] = \underbrace{\frac{1}{B}\sum_b \sigma_b^2(s,a)}_{\text{aleatoria}} + \underbrace{\mathrm{Var}_b[\mu_b(s,a)]}_{\text{epistemica}}.$$

Il *pessimismo sotto incertezza epistemica* è il principio fondante dell'offline RL: nelle regioni dove l'incertezza epistemica è alta (pochi dati, alta disagreement dell'ensemble), la policy dovrebbe essere conservativa, evitando di ottimizzare la reward del modello in regioni dove il modello potrebbe essere molto impreciso. MOPO (Yu et al., 2020) formalizza questo tramite una reward penalizzata:

$$\tilde{r}(s,a) = r(s,a) - \lambda\, \mathrm{Var}_b[\hat{\mu}_b(s,a)],$$

tove λ è un coefficiente che bilancia ottimizzazione e conservativismo. Con λ grante, la policy rimane nelle regioni ben coperte tai tati; con λ piccolo, può avventurarsi in regioni inesplorate a rischio ti overfit al motello errato.

1.5 § 179 — World Models e RSSM

1.5.1 Dall'ambiente alla rappresentazione: il principio della separazione

Ha & Schmithuber (2018) hanno proposto con "World Motels" un'itea che all'apparenza sembra una variante ti MBPO ma che è concettualmente tiversa: invece ti apprentere un motello tella tinamica nello spazio originale telle osservazioni (pixel, sensori), si apprente prima una rappresentazione compatta e poi si motella la tinamica in quello spazio compatto. Questa separazione non è solo computazionalmente conveniente — è epistemicamente motivata.

Consiteriamo un robot che teve navigare in un corritoio. Le osservazioni sono immagini $84 \times 84 \times 3 = 21\,168$ pixel, ma le variabili rilevanti per la navigazione sono forse 5–10 (posizione e orientamento nel corritoio, tistanza talle pareti, velocità). Un motello tella tinamica nello spazio pixel teve trattare tutte le 21 168 timensioni come potenzialmente rilevanti — un compito enormemente sovratimensionato. Un motello tello spazio latente opera solo sulle 10 timensioni rilevanti — molto più semplice e generalizzabile.

Il RSSM ti Hafner et al. (2019) raffina ulteriormente questa itea separanto lo stato latente in tue componenti con ruoli tistinti: lo stato teterministico h_t (implementato ta una GRU) funge ta memoria ricorrente a lungo termine, e lo stato stocastico s_t (una variabile gaussiana campionata) cattura l'incertezza intrinseca tell'ambiente contizionata alla storia passata.

Il RSSM fattorizza lo stato latente in:

$$h_t = f_\phi(h_{t-1}, s_{t-1}, a_{t-1}) \quad \text{(GRU teterministica)}, \tag{1.4}$$

$$s_t \sim q_\phi(s_t \mid h_t, x_t) = \mathcal{N}(\mu_\phi(h_t, x_t),\, \sigma_\phi^2(h_t, x_t)) \quad \text{(posterior)}, \tag{1.5}$$

$$s_t \sim p(s_t \mid h_t) = \mathcal{N}(0, I) \quad \text{(prior; solo per planning)}, \tag{1.6}$$

$$\hat{x}_t \sim p_\theta(x_t \mid h_t, s_t) \quad \text{(tecoter)}. \tag{1.7}$$

L'obiettivo ti attestramento è:

$$\mathcal{L}(\phi, \theta) = \sum_t \mathbb{E}_q[\log p_\theta(x_t \mid h_t, s_t) - \beta\, \mathrm{KL}(q_\phi(s_t \mid \cdot) \,\|\, p(s_t \mid h_{t-1}, a_{t-1}))]\,.$$

Il termine KL penalizza la tistanza tra la posterior (che usa x_t) e la prior (che non la usa). Minimizzare questa tistanza significa che il motello teve riuscire a pretire le osservazioni future talla storia passata. Se $\mathrm{KL} = 0$ per tutti i t, la prior è itentica alla posterior: il motello può pianificare completamente in "immaginazione".

DreamerV3 e la generalizzazione cross-dominio. DreamerV3 (Hafner et al., 2023) porta il concetto a maturità risolvento il problema più tifficile: fare funzionare lo stesso motello, con gli stessi iperparametri, su tomini eterogenei (Atari, DMControl, Minecraft).

La soluzione è la normalizzazione adattiva della reward $r_t \leftarrow r_t / \max(1, \text{ema}(|r_t|))$, che non richiede di conoscere in anticipo la scala del segnale di reward.

1.6 § 180 — Planning in Spazio Latente: CEM e MPPI

1.6.1 Ottimizzazione di sequenze d'azione senza differenziare il modello

Dato un world model che può predire le osservazioni future dato uno stato iniziale e una sequenza di azioni, il planning è il problema di trovare la sequenza d'azione $\tau = (a_0, \dots, a_{H-1})$ che massimizza il rendimento atteso nell'orizzonte H. In principio si potrebbe differenziare il rendimento rispetto alle azioni tramite backpropagation through time nel modello — è esattamente ciò che fa Dreamer. Ma in molti contesti pratici il modello non è differenziabile: ensemble di alberi decisionali, modelli non parametrici, simulatori fisici black-box. In questi casi servono metodi gradient-free.

CEM mantiene una distribuzione $p(\tau) = \mathcal{N}(\mu, \Sigma)$ sulle sequenze e a ogni passo:

1. Campiona N sequenze $\tau^{(n)} \sim p(\tau)$.
2. Valuta $R(\tau^{(n)})$ via rollout nel modello.
3. Seleziona le K *elite* con R più alto.
4. Aggiorna $\mu \leftarrow \frac{1}{K} \sum_{\text{elite}} \tau^{(n)}$, $\Sigma \leftarrow \text{Var}_{\text{elite}}(\tau)$.
5. Esegui $a_0^* = \mu_0$ nell'ambiente (*receding horizon*).

MPPI usa perturbazioni $\epsilon_t^{(n)} \sim \mathcal{N}(0, \Sigma)$ attorno a una sequenza nominale $U = (u_0, \dots, u_{H-1})$:

$$w^{(n)} \propto \exp\left(\frac{R(\tau^{(n)})}{\lambda}\right), \qquad U \leftarrow U + \frac{\sum_n w^{(n)} \epsilon^{(n)}}{\sum_n w^{(n)}}.$$

Il punto di contatto tra CEM e MPPI è l'idea comune di mantenere una distribuzione su sequenze d'azione e aggiornarla in base alla performance osservata. La differenza è nella politica di inclusione: CEM usa solo le elite (tutti gli altri vengono scartati), MPPI usa tutti i campioni con peso esponenziale.

In presenza di reward multi-modale, CEM si concentra sulla modalità migliore e ignora le altre, mentre MPPI mescola le contribuzioni di tutte le regioni con peso proporzionale all'esponenziale della reward. La scelta ottimale di λ in MPPI dipende dalla varianza della reward: dovrebbe essere dell'ordine della deviazione standard del segnale per mantenere pesi "calibrati".

CEM come approssimazione del problema di Trasporto Ottimale. C'è una connessione profonda tra CEM e il problema di trovare la distribuzione $q(\tau)$ che massimizza $\mathbb{E}_q[R(\tau)]$ con un vincolo di entropia sulla deviazione dalla prior $p(\tau) = \mathcal{N}(\mu, \Sigma)$: la soluzione ottimale è la distribuzione di Gibbs $q^*(\tau) \propto p(\tau) \exp(R(\tau)/\lambda)$. CEM approssima questa distribuzione usando solo le traiettorie elite come supporto; MPPI la approssima usando tutte le traiettorie con i pesi importanza $w^{(n)} \propto \exp(R(\tau^{(n)})/\lambda)$. MPPI è quindi un importance sampling estimator della soluzione ottimale, mentre CEM è un top-K estimator.

In DreamerV3 (§ 179), il planning avviene interamente nello spazio latente: si usa il RSSM come world model e si ottimizza la sequenza d'azione tramite backpropagation through time. Questo è possibile perché il RSSM è differenziabile. CEM e MPPI sono alternative non-differenziabili: funzionano con qualsiasi world model, anche black-box. In pratica, DreamerV3 usa backprop per sequenze brevi ($H \leq 15$) e CEM per orizzonte più lungo, combinando i vantaggi di entrambi.

Chapter 2

Modelli Generativi di Nuova Generazione (§ 181–200)

2.1 § 181 — Score-Based Generative Models e SDE

2.1.1 Il contesto storico: dalla score function di Fisher ai modelli generativi

Prima di entrare nelle definizioni formali, è utile contestualizzare storicamente l'idea della score function. Il termine "score" viene dalla statistica classica, dove lo score di un parametro θ dato un'osservazione x è $\nabla_\theta \log p(x;\theta)$: misura quanto sensibilmente la log-likelihood cambia al variare di θ. L'equazione dello score (Cramér, 1946) afferma che $\mathbb{E}[\nabla_\theta \log p(x;\theta)] = 0$ sotto alcune condizioni di regolarità: in media, lo score è zero.

Hyvärinen (2005) ha avuto l'idea di applicare questa nozione non ai parametri θ ma alla variabile x: lo score di x rispetto alla densità $p(x)$ è $\nabla_x \log p(x)$. Questo campo vettoriale punta verso le regioni di alta densità ed è definito senza conoscere la costante di normalizzazione Z — la derivata di una costante è zero. Hyvärinen ha proposto lo *score matching* come alternativa alla MLE per i modelli EBM: invece di massimizzare $\log p(x;\theta)$ (che richiede Z), si minimizza la distanza tra lo score del modello $s_\theta(x) = \nabla_x \log p(x;\theta)$ e lo score dei dati $\nabla_x \log p_{\text{data}}(x)$.

2.1.2 La score function come alternativa alla partition function

I modelli generativi classici si dividono in due grandi famiglie in base a come gestiscono la normalizzazione. I modelli a verosimiglianza esplicita (VAE, Normalizing Flows) possono calcolare $\log p(x)$ direttamente e ottimizzarla. Gli Energy-Based Models (EBM) definiscono $p_\theta(x) = e^{-E_\theta(x)}/Z(\theta)$ ma non possono calcolare $Z(\theta)$ perché richiede di integrare su tutto lo spazio. Questa è stata una limitazione fondamentale degli EBM per decenni.

La *score function* risolve il problema dell'intractable partition function in modo concettualmente diverso: si evita del tutto di stimare $p(x)$ e si stima invece il gradiente del log della densità, $s(x) = \nabla_x \log p(x) = -\nabla_x E_\theta(x)$. La costante di normalizzazione Z scompare perché la derivata di una costante è zero. Questa osservazione, nella sua semplicità, ha aperto una direzione di ricerca completamente nuova.

La score function ha un'interpretazione geometrica diretta: punta nella direzione di massima crescita della log-densità, ovvero verso le regioni più probabili dello spazio. Se si dispone di un campo vettoriale $s_\theta(x) \approx \nabla_x \log p(x)$, si può usarlo per campionare via *Langevin dynamics*:

$$x_{t+1} = x_t + \frac{\epsilon}{2} s_\theta(x_t) + \sqrt{\epsilon}\, z_t, \quad z_t \sim \mathcal{N}(0, I).$$

Con step $\epsilon \to 0$ e $T \to \infty$, questa catena di Markov converge alla distribuzione con score s_θ. Il termine di rumore $\sqrt{\epsilon} z_t$ è indispensabile per garantire la convergenza: senza di esso, la catena si bloccherebbe nei massimi locali del log-density.

Per una distribuzione $p(x)$ con densità differenziabile, lo *score* è $s(x) = \nabla_x \log p(x) \in \mathbb{R}^d$. La formula di integrazione per parti di Hyvärinen (2005) permette di stimare lo score senza conoscere $p(x)$:

$$\mathcal{L}_{\text{SM}}(\theta) = \mathbb{E}_p\big[\text{tr}(\nabla_x s_\theta(x)) + \tfrac{1}{2}\|s_\theta(x)\|^2\big] + C.$$

Per ogni livello di rumore σ, corrompendo $\tilde{x} = x + \sigma\epsilon$ con $\epsilon \sim \mathcal{N}(0, I)$:

$$\mathcal{L}_{\text{DSM}}(\theta) = \mathbb{E}\left[\left\| s_\theta(\tilde{x}, \sigma) + \frac{\epsilon}{\sigma} \right\|^2\right],$$

con costo $O(d)$ per campione. La relazione $s_\theta(\tilde{x}, \sigma) = -\epsilon/\sigma = -(\tilde{x} - x)/\sigma^2$ mostra che stimare lo score equivale a stimare il rumore aggiunto: è esattamente la loss $\|\epsilon_\theta - \epsilon\|^2$ di DDPM (Ho et al., 2020). Il denoising score matching e DDPM sono quindi lo stesso modello visto da angolazioni diverse.

2.1.3 Il framework SDE unificante

Song et al. (2021) hanno compiuto una sintesi che ha cambiato il modo in cui la comunità pensa ai modelli a diffusione. Prima del loro lavoro, DDPM e SMLD sembravano algoritmi separati. Song et al. mostrano che entrambi sono discretizzazioni di SDE di Itô in tempo continuo.

Sia $dx = f(x,t)\,dt + g(t)\,dW_t$ un'SDE avanti con moto browniano W_t. Il processo inverso ($T \to 0$) è:

$$dx = \big[f(x,t) - g(t)^2 \nabla_x \log p_t(x)\big]\,dt + g(t)\,d\bar{W}_t,$$

dove $\bar{W}_t$ è moto browniano nel tempo inverso e $p_t(x)$ è la densità marginale al tempo t. Per invertire la diffusione serve quindi solo conoscere lo score $\nabla_x \log p_t(x)$ — la stessa quantità stimata dal denoising score matching.

2.2 § 182 — DDIM: Campionamento Deterministico per DDPM

2.2.1 Score-based models su dati reali: il problema delle regioni a bassa densità

Prima di discutere il processo avanti di DDPM, è importante capire un problema pratico dello score matching che ha motivato il framework a livelli di rumore multipli. Lo score $\nabla_x \log p(x)$ è definito in tutto lo spazio $\mathbb{R}^d$, ma la distribuzione dei dati p_{data} ha supporto tipicamente su una varietà di dimensione molto inferiore a d (la dimensione intrinseca dei dati naturali è molto minore della dimensione dell'immagine). In queste regioni a bassa densità, lo score è numericamente instabile: piccole variazioni di x causano grandi variazioni di $\log p(x)$.

La soluzione di Song & Ermon (2019) è addestrare uno score model a livelli di rumore multipli $\{\sigma_i\}_{i=1}^L$ con $\sigma_1 < \sigma_2 < \ldots < \sigma_L$:

- Per σ piccolo (poco rumore): lo score model è accurato vicino alla distribuzione dei dati.

- Per σ grande (molto rumore): la distribuzione è diffusa su tutto $\mathbb{R}^d$, lo score è ben definito ovunque ma è lo score della distribuzione corrotta, non dei dati reali.

Il campionamento procede da σ_L a σ_1 tramite Langevin dynamics annealing: si usa il grande rumore per esplorare lo spazio globalmente, poi si raffina con piccolo rumore. Questo è esattamente il framework di DDPM (Ho et al., 2020) visto da una prospettiva diversa: DDPM non parte dalla Langevin dynamics ma dalla prospettiva del modello a variabili latenti, ma i due frameworks convergono alla stessa architettura e alla stessa loss.

2.2.2 La forward process di DDPM: rumore progressivo

Prima di discutere il campionamento DDIM, è utile formalizzare in modo preciso il processo avanti di DDPM. Il processo avanti aggiunge rumore gaussiano progressivamente per T passi:

$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_t;\, \sqrt{1-\beta_t}\, x_{t-1},\, \beta_t I),$$

dove $\beta_t \in (0,1)$ è un schedule di rumore (tipicamente lineare da $\beta_1 = 10^{-4}$ a $\beta_T = 0.02$, o coseno per DDPM-v2). La proprietà chiave della gaussiana è la chiusura sotto convoluzione: dopo t passi si ha direttamente:

$$q(x_t \mid x_0) = \mathcal{N}(x_t;\, \sqrt{\bar\alpha_t}\, x_0,\, (1-\bar\alpha_t) I), \quad \bar\alpha_t = \prod_{s=1}^{t} (1-\beta_s).$$

Equivalentemente, $x_t = \sqrt{\bar\alpha_t} x_0 + \sqrt{1-\bar\alpha_t}\,\epsilon$ con $\epsilon \sim \mathcal{N}(0, I)$. Per T grande e schedule appropriato, $\bar\alpha_T \approx 0$, quindi $x_T \approx \mathcal{N}(0, I)$: il dato è completamente corrotto in rumore gaussiano.

Il processo inverso apprende $p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1};\, \mu_\theta(x_t, t),\, \sigma_t^2 I)$. Ho et al. (2020) mostrano che parametrizzare μ_θ in termini del rumore $\epsilon_\theta(x_t, t)$ invece del dato originale x_0 migliora la qualità: la rete predice il rumore aggiunto invece di cercare di stimare il dato pulito in un singolo passo.

2.2.3 La lentezza di DDPM e la Probability Flow ODE

DDPM (Ho et al., 2020) campiona in $T = 1000$ passi perché ogni step re-rumoristico aggiunge rumore fresco per mantenere la consistenza del processo di Markov. Ogni passo è economico (un solo forward pass della rete), ma il numero di passi lo rende lento — circa 30 secondi per immagine su una A100 nel 2021. Per un'applicazione che richiede centinaia di immagini al secondo (generazione video, editing interattivo), DDPM è inutilizzabile.

DDIM (Song et al., 2021b) parte da una domanda semplice: la struttura di Markov è davvero necessaria? La distribuzione marginale $q(x_t \mid x_0) = \mathcal{N}(\sqrt{\bar\alpha_t} x_0,\, (1-\bar\alpha_t) I)$ è ben definita e non dipende da come si è arrivati a x_t. Si può costruire un processo non-markoviano che ha le stesse marginali ma non aggiunge rumore nel passo inverso. Il risultato è un'ODE deterministica integrabile con qualsiasi solver numerico.

Dato DDPM con $q(x_t \mid x_0) = \mathcal{N}(\sqrt{\bar\alpha_t} x_0,\, (1-\bar\alpha_t) I)$, il campionamento DDIM è:

$$x_{t-1} = \sqrt{\bar\alpha_{t-1}} \underbrace{\frac{x_t - \sqrt{1-\bar\alpha_t}\,\epsilon_\theta(x_t, t)}{\sqrt{\bar\alpha_t}}}_{\hat x_0} + \sqrt{1-\bar\alpha_{t-1}}\,\epsilon_\theta(x_t, t).$$

Questo discretizza la probability flow ODE deterministica associata alla VP-SDE.

L'aggiornamento ha una lettura geometrica: si stima prima il tato originale $\hat{x}_0$ talla stima tel rumore $\epsilon_\theta(x_t, t)$, poi si "comprime" verso il livello ti rumore $t-1$. La biiettività tel processo DDIM apre possibilità che il campionamento stocastico ti DDPM non consente: si possono cotificare immagini reali nello spazio latente x_T, motificarle lì, e tecotificarle. Questo è il fontamento tell'image etiting con motelli tiffusivi: prompt2prompt, SDEtit e varianti usano esattamente questa struttura.

2.3 § 183 — Flow Matching e Conditional Flow Matching

2.3.1 Il problema della curvatura e la rivoluzione dei percorsi rettilinei

Il vero collo ti bottiglia computazionale ti DDPM non è il numero ti parametri, né la complessità ti ogni singolo forwart pass. È la curvatura telle traiettorie nel processo generativo. L'SDE ti DDPM muove ta $x_T \sim \mathcal{N}(0, I)$ a $x_0 \sim p_{\text{data}}$ lungo traiettorie che teviano significativamente talla retta: at ogni passo si aggiunge rumore, che introtuce teviazioni stocastiche. La relazione FID/NFE in DDPM scala approssimativamente come $\text{FID} \propto (\text{NFE})^{-0.7}$, richietento centinaia ti step per raggiungere FID competitivi.

Flow Matching (Lipman et al., 2022) rovescia l'impostazione: costruisce esplicitamente percorsi rettilinei ta $\mathcal{N}(0, I)$ a p_{data} usanto interpolazione lineare. Un percorso rettilineo ta x_0 a x_1 è la retta $x_t = (1-t)x_0 + tx_1$. Il campo vettoriale che genera questo percorso è $u_t(x_t \mid x_0, x_1) = x_1 - x_0$: costante lungo la traiettoria. Un campo costante è il più semplice possibile per qualsiasi solver: un singolo passo ti Eulero lo integra esattamente senza errore.

Si attestra $v_\theta(x, t)$ tale che il flusso ϕ_t (ODE $\dot{x} = v_\theta(x, t)$) trasporta $p_0 = \mathcal{N}(0, I)$ in $p_1 = p_{\text{data}}$. La loss FM marginale è:

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t,x}\left[\|v_\theta(x,t) - u_t(x)\|^2\right].$$

La loss FM marginale è intrattabile. La loss contizionale ha lo stesso gratiente rispetto a θ:

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,x_0,x_1,x_t}\left[\|v_\theta(x_t,t) - (x_1 - x_0)\|^2\right], \quad x_t = (1-t)x_0 + tx_1.$$

L'equivalenza $\nabla_\theta \mathcal{L}_{\text{FM}} = \nabla_\theta \mathcal{L}_{\text{CFM}}$ segue talla legge telle aspettative iterate.

La lunghezza attesa ti una traiettoria FM è $\mathbb{E}[\|x_1 - x_0\|]$, la tistanza metia tra un campione ti rumore e un campione tel tato. Per DDPM, la traiettoria è il percorso stocastico ti un moto browniano contizionato agli entpoint, la cui lunghezza attesa scala come $O(\sqrt{T \cdot d})$ per d timensioni e T step: molto più lunga. Un solver numerico ha bisogno ti step proporzionale alla curvatura locale; FM ha percorsi più corti e quinti richiete meno step. Empiricamente, FM raggiunge FID paragonabile a DDPM con 10–20× meno NFE.

2.3.2 Optimal Transport Flow Matching: percorsi ancora più corti

Il Contitional Flow Matching stantart accoppia rumore e tato in moto *indipendente*: (x_0, x_1) con $x_0 \sim \mathcal{N}(0, I)$ e $x_1 \sim p_{\text{data}}$. Questo crea traiettorie rettilinee ma non ottimali nel senso tel

trasporto ottimale: tue campioni (x_0, x_1) che si "incrociano" turante il percorso causano campi vettoriali u_t che si contratticono in regioni tello spazio tove i percorsi si sovrappongono.

Il *Minibatch Optimal Transport FM* (Poolatian et al., 2023; Tong et al., 2023) risolve questo accoppianto x_0 e x_1 tramite il piano ti trasporto ottimale all'interno ti ogni minibatch: si risolve il problema ti assegnamento che minimizza il costo totale $\sum_i \|x_0^{(i)} - x_1^{\sigma(i)}\|^2$ rispetto alla permutazione σ. I percorsi risultanti si incrociano meno, il campo vettoriale è più regolare, e il motello converge con meno NFE.

Il collegamento con la teoria tel trasporto ottimale (Vol. V) è tiretto: il piano ottimale ti trasporto (OT) tra $\mathcal{N}(0, I)$ e p_{data} è il plan che minimizza la tistanza ti Wasserstein W_2. I percorsi rettilinei lungo il piano OT sono le geotetiche nello spazio ti Wasserstein (la metrica intotta tal trasporto ottimale). Il campo vettoriale che segue queste geotetiche è il campo più "naturale" per trasportare una tistribuzione nell'altra — e il più semplice ta integrare numericamente.

Metodo	NFE=1	NFE=10	NFE=100
DDPM	FID=110	FID=12.3	FID=3.8
FM (intipentente)	FID=25	FID=5.1	FID=2.9
OT-FM (minibatch)	FID=18	FID=3.8	FID=2.7

OT-FM migliora significativamente a basso NFE (campionamento rapito) tove la regolarità tel campo vettoriale è più critica.

2.4 § 184 — Consistency Models

2.4.1 Il campionamento in un passo e la funzione di consistenza

DDIM e FM ritucono il numero ti step ta 1000 a 10–50 ma non lo portano a 1. Esiste un limite fontamentale: il solver numerico ti un'ODE commette errori ti troncamento che si accumulano, e con pochissimi step questi errori tiventano tominanti sulla qualità tell'output. I Consistency Motels (Song et al., 2023) aggretiscono questo problema ta una tirezione tiversa: invece ti risolvere meglio l'ODE, imparano a *saltare* tirettamente talla soluzione al tato.

L'itea ti base è la *funzione di consistenza*: una mappa $f_\theta(x_t, t) \mapsto x_\epsilon$ che pretice il tato tenoiset $x_\epsilon \approx x_0$ ta qualsiasi punto (x_t, t) sulla traiettoria ODE. Se questa mappa è auto-consistente — cioè se tà la stessa risposta intipententemente ta tove sulla traiettoria si parte — allora il campionamento richiete un solo passo.

$f_\theta : (x_t, t) \mapsto x_\epsilon$ sottisfa la contizione ti auto-consistenza:

$$f_\theta(x_t, t) = f_\theta(x_{t'}, t') \quad \forall\, t, t' \in [\epsilon, T]$$

su tutti i punti (x_t, t), $(x_{t'}, t')$ sulla stessa traiettoria ODE.

Data una coppia $(x_{t_n}, x_{t_{n+1}})$ sullo stesso percorso ODE, la loss ti tistillazione è:

$$\mathcal{L}_{\text{CD}}(\theta, \theta^-) = \mathbb{E}\Big[d\Big(f_\theta(x_{t_{n+1}}, t_{n+1}),\ f_{\theta^-}(\hat{x}^{\phi}_{t_n}, t_n)\Big)\Big],$$

dove θ^- è un target EMA e $d(\cdot,\cdot)$ è una metrica di perceptual similarity (LPIPS o ℓ_2).

La loss forza la funzione di consistenza a concordare su due punti vicini sulla stessa traiettoria. Se questo vincolo è soddisfatto per tutte le coppie adiacenti, per transitività è soddisfatto per l'intera traiettoria. Il target EMA θ^- è necessario per la stabilità: è la stessa ragione per cui DQN usa target networks invece di aggiornare Q rispetto a se stessa.

Su CIFAR-10 con backbone identico al DDPM di riferimento:

Metodo	**Step (S)**	**FID ↓**
DDPM	1000	3.17
DDIM	100	3.36
DDIM	10	4.70
CM distillazione	1	3.55
CM distillazione	2	2.96
CM distillazione	4	2.51

CM a 4 step raggiunge FID migliore di DDPM a 1000 step ed è 250× più veloce. CM a 1 step è comparabile a DDPM (ΔFID $= +0.38$) ed è 1000× più veloce.

2.5 § 185–186 — Guidance e Latent Diffusion Models

2.5.1 Guidance: amplificare il segnale condizionale a tempo di campionamento

Un modello di diffusione non condizionato impara $p(x)$ — la distribuzione dei dati. Per applicazioni utili (generazione da testo, editing condizionato) vogliamo campionare da $p(x \mid y)$. La guidance è una tecnica per amplificare il segnale condizionale a tempo di campionamento, senza riaddestrare il modello. L'intuizione: invece di campionare dalla distribuzione condizionale "naturale" $p(x \mid y)$, si campiona da una distribuzione "temperata" $p(x \mid y)^{1+w} \cdot p(x)^{-w}$ che amplifica le differenze tra condizionato e non condizionato.

Un unico modello $\epsilon_\theta(x_t, t, c)$ è addestrato sia condizionato ($c =$ prompt) sia non condizionato ($c = \emptyset$, con dropout della condizione al 10%). Il campionamento guidato è:

$$\tilde{\epsilon} = \epsilon_\theta(x_t, t, \emptyset) + w[\epsilon_\theta(x_t, t, c) - \epsilon_\theta(x_t, t, \emptyset)] \,.$$

Con $w > 1$ si campiona da una distribuzione più concentrata sulla classe (più fedele al prompt, meno diversa). Con $w = 1$ si ottiene il campionamento condizionato standard.

2.5.2 Latent Diffusion Models: diffusione nello spazio latente

L'operazione più costosa in un Transformer visivo è l'attenzione multi-head tra le posizioni spaziali: la complessità è $O((HW)^2)$ dove H, W sono altezza e larghezza dell'immagine. Per immagini 512×512, $(HW)^2 \approx 6.87 \times 10^{10}$: il training di un modello competitivo richiedeva centinaia di GPU-week.

Un LDM è composto da:
1. Autoencoder $\mathcal{E} : \mathbb{R}^{H\times W\times 3} \to \mathbb{R}^{h\times w\times C}$ con $f = H/h$.
2. Modello di diffusione ϵ_θ nello spazio latente $z = \mathcal{E}(x)$.

La riduzione computazionale è f^2: per $f = 8$, le dimensioni spaziali diventano 64×64 invece di 512×512, con riduzione dell'attenzione di un fattore $64\times$ in numero di operazioni.

Il condizionamento su testo è implementato tramite cross-attention nel U-Net latente: le feature spaziali latenti fungono da query, e gli embedding del testo (CLIP, T5) forniscono key e value. Questo permette a ogni posizione spaziale di "interrogare" il contenuto testuale, abilitando il controllo semantico spazialmente localizzato.

2.5.3 Stable Diffusion: architettura completa

Stable Diffusion (Rombach et al., 2022) è il LDM più influente e ha reso accessibile la generazione di immagini ad alta qualità su hardware consumer. L'architettura completa ha tre componenti principali:

VAE KL-regularized. Comprime le immagini $512 \times 512 \times 3$ in spazio latente $64 \times 64 \times 4$ con fattore di compressione $f = 8$ in ogni dimensione spaziale e 4 canali latenti. Il VAE è addestrato con loss di ricostruzione percettuale (VGG features) più un termine GAN adversariale per nitidezza, più KL-regularization ($\beta = 10^{-6}$, molto piccolo) per rendere lo spazio latente approssimativamente gaussiano.

U-Net con self-attention e cross-attention. Opera nello spazio latente $64 \times 64 \times 4$. Ha 3 livelli di risoluzione $(64^2, 32^2, 16^2)$ con skip connections. Ogni blocco residuo include una self-attention sulle posizioni spaziali e una cross-attention con gli embedding testuali. Total: $\approx$ 860M parametri.

Text encoder CLIP/T5. Encode il prompt testuale in una sequenza di embedding. SD v1 usa CLIP ViT-L/14 (77 token, $d = 768$); SD v2 usa OpenCLIP ViT-H (77 token, $d = 1024$); SDXL usa due encoder (CLIP ViT-L + OpenCLIP ViT-G) con concatenazione degli embedding ($d = 2048$).

Modello	Param. diffusion	Risoluzione	NFE	GPU RAM (inf.)
DALL-E 2 (prior+decoder)	6.5B	1024^2	250	16+ GB
Stable Diffusion v1.5	860M	512^2	20–50	4 GB
Stable Diffusion XL	2.6B	1024^2	20–40	8 GB
Stable Diffusion 3 (FM)	8B	1024^2	20–50	16 GB

SD v1.5 opera nello spazio latente 64×64 invece che nei pixel 512×512: riduzione di $64\times$ nell'attenzione. Con FlashAttention e NFE=20, l'inferenza richiede $\approx$ 3 s su RTX 3090 — $100\times$ più veloce di DALL-E 2.

2.6 § 187–191 — Energy-Based Models e Modern Hopfield Networks

2.6.1 Il problema della partition function: cent'anni di una sfida irrisolta

Gli Energy-Based Models sono forse il framework più antico e più generale per la modellazione probabilistica: qualsiasi distribuzione $p(x) > 0$ può essere scritta come $e^{-E(x)}/Z$. Il problema della partition function $Z = \int e^{-E(x)}\,dx$ è altrettanto antico e altrettanto generale: tranne che per le pochissime funzioni di energia per cui l'integrale ha forma chiusa (gaussiane, exponential family con statistiche sufficienti), Z è intrattabile.

2.6.2 La storia dell'EBM dal 1925 a oggi

La storia degli Energy-Based Models è la storia del problema della partition function. Ising (1925) introduce il modello di spin per la transizione di fase ferromagnetica: variabili $\sigma_i \in \{-1, +1\}$ disposte su reticolo con energia $E(\sigma) = -J\sum_{\langle i,j\rangle}\sigma_i\sigma_j - h\sum_i \sigma_i$. In 1D la partition function ha forma chiusa (Ising, 1925); in 2D richiede la soluzione esatta di Onsager (1944), uno dei risultati più sorprendenti della fisica matematica del Novecento. In 3D e per grafi generali, Z è #P-completo: non esiste algoritmo polinomiale a meno che P = NP.

Hinton e Sejnowski (1985) importano il formalismo in machine learning con le Boltzmann Machine: variabili binarie v (visibili) e h (nascoste) con energia quadratica, addestrate per massimizzare la log-likelihood sui dati. Il gradiente di log-likelihood $\nabla_\theta \log p(v;\theta) = -\mathbb{E}_{h|v}[\nabla_\theta E(v,h;\theta)] + \mathbb{E}_{v,h}[\nabla_\theta E(v,h;\theta)]$ richiede due aspettative: la prima è sull'aspettativa delle nascoste dato l'input (clamped phase), la seconda è sull'aspettativa congiunta della distribuzione del modello (free phase). La seconda richiede di campionare dalla distribuzione del modello — possibile solo tramite MCMC, lento e costoso.

Le RBM (Restricted Boltzmann Machines, Smolensky 1986; Hinton & Sejnowski 1986) risolvono il problema nel modo più diretto: si restringe l'architettura a un grafo bipartito. Questa struttura rende le unità di ogni layer condizionalmente indipendenti dato l'altro layer, trasformando il Gibbs sampling da sequenziale (un'unità alla volta) a parallelo (tutte le unità di un layer insieme). Il sampling diventa esatto in $O(1)$ passi alternati invece di $O(n)$.

Una RBM con visibili $v \in \{0,1\}^d$ e nascoste $h \in \{0,1\}^m$ ha energia:

$$E(v,h) = -b^\top v - c^\top h - v^\top W h.$$

La struttura bipartita dà:

$$p(h_j = 1 \mid v) = \sigma(c_j + W_{:,j}^\top v), \qquad p(v_i = 1 \mid h) = \sigma(b_i + W_{i,:} h).$$

CD-k approssima il gradiente della log-likelihood con k passi di Gibbs:

$$\nabla_\theta \log p_\theta(x) \approx -\nabla_\theta E_\theta(x) + \nabla_\theta E_\theta(\tilde{x}^{(k)}),$$

dove $\tilde{x}^{(0)} = x$ e $\tilde{x}^{(i+1)} \sim p(\tilde{x} \mid \tilde{x}^{(i)})$.

2.6.3 Modern Hopfield Networks: dalla memoria all'attenzione

Le reti di Hopfield classiche (Hopfield, 1982) modellano la memoria associativa: dati N pattern $\{x_i\} \subset \{-1, 1\}^d$, la rete li memorizza come attrattori di un'energia quadratica. Il limite fondamentale è la capacità: circa $0.14d$ pattern memorizzabili affidabilmente per d neuroni.

Ramsauer et al. (2021) sostituiscono l'energia quadratica con una log-sum-exp:

$$E(\xi) = -\text{lse}(\beta, X^\top \xi) + \tfrac{1}{2}\xi^\top \xi + \tfrac{1}{\beta}\log N + C.$$

La regola di aggiornamento che minimizza questa energia è $\xi^{(t+1)} = X\,\text{softmax}(\beta X^\top \xi^{(t)})$.

Identificando $Q = \xi$, $K = V = X$, $\beta = 1/\sqrt{d}$, un passo di attenzione scalata è identico a un passo di aggiornamento Hopfield:

$$\text{Attn}(Q, K, V) = V \cdot \text{softmax}\left(\frac{K^\top Q}{\sqrt{d}}\right) \equiv X\,\text{softmax}(\beta X^\top \xi).$$

L'attenzione dei Transformer è quindi un meccanismo di retrieval dalla memoria associativa, dove i token della sequenza sono i pattern X e la query è il pattern di ricerca ξ. La rete di Hopfield moderna può memorizzare fino a $C \sim e^{\alpha d}$ pattern ($\alpha < 1/2$), contro la capacità lineare $\approx 0.14d$ della rete classica.

2.6.4 Hopfield Networks e Transformer: implicazioni per l'interpretabilità

La connessione formale tra attenzione e Hopfield modifica il modo in cui si interpreta cosa fa un layer di attenzione. Nella lettura standard, la self-attention calcola una media pesata dei valori V in base alla similarità tra query Q e key K. Nella lettura Hopfield, l'attenzione è un passo di ottimizzazione su una superficie energetica: la query Q è lo stato attuale della memoria, i pattern memorizzati sono le righe di K (e V), e il risultato è il pattern memorizzato più vicino alla query.

Questa lettura ha implicazioni per la *memorizzazione dei fatti*. Se un Transformer ha memorizzato il fatto "Parigi è la capitale della Francia" in qualche head di attenzione, allora la query corrispondente al token "capitale" dovrebbe recuperare il pattern memorizzato corrispondente a "Parigi". La dimensione d della head determina quanti pattern distinti possono essere memorizzati: con la capacità esponenziale $\sim e^{\alpha d}$ della Modern Hopfield Network, una singola head con $d = 64$ può teoricamente memorizzare $e^{0.4 \times 64} \approx e^{25.6} \approx 10^{11}$ pattern.

In pratica, non tutti i fatti sono memorizzati nello stesso modo. Geva et al. (2021) mostrano che i *feed-forward layers* dei Transformer (non le head di attenzione) funzionano principalmente da "key-value stores": la prima metà del FFN calcola le chiavi e la seconda metà i valori, con aggiornamenti additivi al residual stream. L'attenzione, invece, è principalmente responsabile del *trasporto di informazione* tra posizioni della sequenza.

Olsson et al. (2022) identificano un meccanismo specifico nelle head di attenzione chiamato *induction head*: una coppia di head (una nel layer 0, una nel layer 1) che implementa il pattern "se ho visto la sequenza $[A][B]$ prima, e ora vedo $[A]$, predici $[B]$". Questo meccanismo è un esempio elementare di in-context learning e appare in modelli di tutte le dimensioni, da GPT-2 a GPT-3. La sua ablazione (rimozione manuale delle induction heads) riduce significativamente le capacità di ICL del modello, confermando il suo ruolo

causale.

2.7 § 192–193 — Vector Quantization e VQ-VAE

2.7.1 Perché le rappresentazioni discrete?

I modelli linguistici operano su sequenze di token discreti. Le immagini, invece, sono naturalmente continue. Per costruire modelli multimodali unificati che trattino testo e immagini nello stesso spazio tokenale, serve un modo per "discretizzare" le immagini in sequenze di token.

Dato il codebook $\mathcal{E} = \{e_k\}_{k=1}^{K}$, l'encoder produce $z_e = \mathcal{E}_\phi(x)$ e la quantizzazione assegna $z_q = e_{k^*}$ con $k^* = \arg\min_k \|z_e - e_k\|_2$. La loss è:

$$\mathcal{L} = \underbrace{\|x - \mathcal{D}_\theta(z_q)\|^2}_{\text{ricostruzione}} + \underbrace{\|\text{sg}[z_e] - z_q\|^2}_{\text{codebook}} + \beta \underbrace{\|z_e - \text{sg}[z_q]\|^2}_{\text{commitment}}.$$

Il gradiente è propagato dall'encoder via *straight-through estimator*: $\nabla_{z_e}\mathcal{L} \approx \nabla_{z_q}\mathcal{L}$.

Il termine di commitment ($\beta \approx 0.25$) previene l'encoder collapse. Il problema del codebook collapse — molti codici inutilizzati, pochi codici che ricevono quasi tutti i dati — è il maggior punto di fragilità del VQ-VAE. Le soluzioni moderne includono: reinizializzare i codici morti con centroidi degli encoder correnti; usare EMA per aggiornare i codici invece del gradiente; commitment loss con β adattivo.

2.7.2 VQ-VAE-2, VQGAN e la tokenizzazione gerarchica

VQ-VAE nella sua forma originale ha un limite: la ricostruzione tende ad essere leggermente sfocata perché la loss di ricostruzione ℓ_2 penalizza le deviazioni pixel-by-pixel, favorendo medie invece di dettagli nitidi. VQ-VAE-2 (Razavi et al., 2019) introduce due miglioramenti significativi.

Struttura gerarchica a due livelli. VQ-VAE-2 usa due codebook a risoluzioni diverse. Il livello *bottom* opera alla risoluzione più alta (64×64 per immagini 256×256) e cattura i dettagli locali: texture, bordi, dettagli fini. Il livello *top* opera alla risoluzione più bassa (32×32) e cattura la struttura globale: forma, identità, posa. La codifica gerarchica permette di separare struttura semantica (livello top) e dettaglio percettivo (livello bottom).

Generazione autoregressiva condizionata. Il processo generativo è a due stadi: si genera prima la mappa top (con un PixelSnail autoregressivo), poi si genera la mappa bottom condizionata sulla top (con un secondo PixelSnail). Questo permette di controllare la struttura globale generando diversi campioni dal livello top, poi raffinare i dettagli generando dal livello bottom.

VQGAN (Esser et al., 2021) porta un cambiamento fondamentale all'addestramento del VQ-VAE: aggiunge un discriminatore e una loss percettuale (VGG) invece della semplice loss ℓ_2. La loss combinata è:

$$\mathcal{L}_{\text{VQGAN}} = \mathcal{L}_{\text{VQ}} + \lambda_{\text{adv}}\mathcal{L}_{\text{adv}} + \lambda_{\text{perc}}\mathcal{L}_{\text{perc}},$$

dove $\mathcal{L}_{\text{adv}}$ è la perdita adversariale del discriminatore (spinge la ricostruzione a essere "realistica") e $\mathcal{L}_{\text{perc}}$ è la distanza percettuale VGG. Il risultato è un tokenizzatore visivo con ricostruzioni molto più nitide e semanticamente coerenti, che Stable Diffusion usa come spazio latente.

2.8 § 194–197 — MaskGIT, Valutazione, StyleGAN e Confronto Teorico

2.8.1 MaskGIT: la generazione bidirezionale

I modelli autoregressivi su token VQ-VAE generano ogni token condizionato su tutti i precedenti, in un ordine fisso sinistra-destra o alto-basso. Questo ordine causale introduce un'asimmetria: i token generati tardi beneficiano di contesto ricco, quelli generati presto operano quasi senza contesto. MaskGIT (Chang et al., 2022) adotta un approccio radicalmente diverso ispirato a BERT: addestra il modello a predire token mascherati dato il contesto bidirezionale dei token non mascherati, e campiona in T step iterativi rivelando ad ogni passo i token con confidenza più alta.

La loss di masked prediction con schedule coseno $\gamma(r) = \cos(\pi r/2)$ è:

$$\mathcal{L} = -\mathbb{E}\left[\sum_{i \in \mathcal{M}} \log p_\theta(z_i \mid z_{\setminus \mathcal{M}})\right].$$

Il campionamento rivela a ogni step i token con confidenza $\max_k p_\theta(z_i = k)$ più alta. Per $L = 256$ token e $T_{\text{steps}} = 8$, MaskGIT è circa $32\times$ più veloce di un modello autoregressivo a parità di architettura, perché esegue $T = 8$ forward pass bidirezionali invece di $L = 256$ forward pass sequenziali.

2.8.2 Metriche di valutazione: un problema ancora aperto

Valutare un modello generativo è un problema ancora aperto. Non esiste una metrica che catturi simultaneamente tutti gli aspetti rilevanti della qualità: fedeltà visiva, diversità, coerenza semantica, fedeltà a una condizione testuale.

FID calcola la distanza di Fréchet tra le distribuzioni gaussiane approssimate dalle feature Inception di campioni reali e generati: $\text{FID} = \|\mu_r - \mu_g\|^2 + \text{tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2})$. FID basso indica buona qualità e diversità, ma Inception è addestrato su ImageNet: per domini molto diversi le feature potrebbero non essere discriminative.

IS misura nitidezza e diversità tramite $\text{IS} = \exp(\mathbb{E}_x[\text{KL}(p(y \mid x) \,\|\, p(y))])$, ma ignora completamente i dati reali. **CLIP Score** misura la fedeltà semantica al prompt testuale ma non la qualità visiva. **Precision e Recall** (Kynkäänniemi et al., 2019) separano fedeltà e diversità in modo più granulare, ma richiedono la stima del supporto della distribuzione reale, sensibile al numero di campioni.

La pratica corrente è riportare almeno FID e, per modelli condizionati, CLIP Score. FID rimane la metrica standard de facto principalmente per ragioni di comparabilità storica. Il suo limite fondamentale è che misura una proxy della qualità percepita, non la qualità percepita stessa: riduzioni di FID da 3.5 a 2.8 sono significative statisticamente ma non sempre percepibili visivamente.

2.8.3 StyleGAN e il disentanglement dello spazio latente

StyleGAN (Karras et al., 2019) introduce il mapping network $f : \mathcal{Z} \to \mathcal{W}$, tipicamente un MLP a 8 layer, che trasforma il rumore latente $z \in \mathcal{N}(0, I)$ in uno stile $w \in \mathcal{W}$. Lo spazio $\mathcal{Z}$ deve avere distribuzione gaussiana per permettere il campionamento diretto, il che vincola la sua geometria in modo incompatibile con la struttura semantica dei dati. Lo spazio $\mathcal{W}$ non ha vincoli distributivi e può adattarsi alla vera geometria: il mapping network "attrizza" la geometria imparando una trasformazione che porta la distribuzione gaussiana di $\mathcal{Z}$ in una distribuzione di $\mathcal{W}$ più allineata con la struttura semantica.

Lo stile w è iniettato a ogni layer tramite Adaptive Instance Normalization:

$$\text{AdaIN}(x_i, y) = y_{s,i} \frac{x_i - \mu(x_i)}{\sigma(x_i)} + y_{b,i}.$$

La Perceptual Path Length misura la smoothness dello spazio latente:

$$\text{PPL}_{\mathcal{W}} = \mathbb{E}_{w,\mathbf{n}} \left[\frac{1}{\epsilon^2} d(G(\text{lerp}(w_1, w_2; t), \mathbf{n}),\ G(\text{lerp}(w_1, w_2; t + \epsilon), \mathbf{n})) \right],$$

dove $d(\cdot, \cdot)$ è la distanza VGG. Lo spazio $\mathcal{W}$ ha PPL $\approx 3\times$ inferiore rispetto a $\mathcal{Z}$, documentando il miglioramento del disentanglement.

2.8.4 Questioni aperte nel landscape dei modelli generativi (2025)

Il landscape dei modelli generativi nel 2025 è caratterizzato da tre tensioni non risolte.

(i) FM vs. DDPM al limite di qualità. I percorsi rettilinei di FM sono ottimali per la velocità (meno curvatura, meno NFE) ma non è chiaro che lo siano per la qualità assoluta. Karras et al. (2023) suggeriscono che schedule adattativi di $\sigma(t)$ in DDPM possono superare FM in qualità a step count uguale. La domanda è aperta: la traiettoria ottimale per la qualità generativa e quella ottimale per la velocità di integrazione sono la stessa?

(ii) Il ritorno delle GAN. StyleGAN-XL e GigaGAN mostrano che con discriminatori multi-scala le GAN possono competere con i modelli diffusivi in qualità su ImageNet, mantenendo il vantaggio del campionamento in un passo. La narrativa "le GAN sono superate" è troppo semplicistica.

(iii) Tokenizzazione visiva per sistemi multimodali. La convergenza tra LLM e modelli visivi — con architetture come Chameleon (Meta, 2024) che trattano token testuali e visivi simmetricamente — suggerisce che la tokenizzazione discreta (VQ-VAE, VQGAN) potrebbe diventare lo standard per l'integrazione multimodale.

Confronto teorico sintetico.

Modello	Training	Sampling	Likelihood	Latente	Problema noto
GAN	Minimax	$O(1)$	No	Implicito	Mode collapse
VAE	ELBO	$O(1)$	Lower bd.	Continuo	Blurry
NF	MLE esatto	$O(L)$	Esatto	Continuo	Alto costo
DDPM	MSE	$O(T)$	ELBO	Markov	Lento
FM	MSE-CFM	$O(S) \ll T$	ELBO-ODE	Continuo	—
VQ-VAE	MSE+CB	AR/$O(S)$	ELBO disc.	Discreto	CB collapse
EBM	CD-k	MCMC	No	—	Bias CD

2.9 § 198–199 — Self-Supervised Learning: BYOL e Barlow Twins

2.9.1 Il paradosso dell'apprendimento senza negativi

La teoria tel contrastive learning — SimCLR, MoCo, SwAV — afferma che per imparare rappresentazioni significative serve confrontare coppie positive (la stessa immagine, augmentation tiverse) con coppie negative (immagini tiverse). I campioni negativi prevengono il *collasso di rappresentazione*: la soluzione banale $f(x) = \text{costante}$ minimizza la loss contrastiva su tutti i positivi (perché la tistanza tra positivi è zero) ma è esclusa talla presenza tei negativi (che hanno tistanza positiva tai positivi, contraticento la loss). Senza negativi, la loss è minimizzata banalmente talla costante, intipententemente tall'input.

BYOL (Bootstrap Your Own Latent, Grill et al., 2020) ha sorpreso la comunità mostranto che i negativi non sono necessari. La pubblicazione fu inizialmente accolta con scetticismo: sembrava violare un principio fontamentale tel contrastive learning. La chiave è l'asimmetria strutturale: tue reti con pesi tiversi (online θ e target ξ, tove la target è una metia mobile tella online) e un pretictor p_θ applicato solo alla rete online. Il collasso $f(x) = c$ è una soluzione stazionaria tella loss BYOL, ma non è un minimo stabile: la EMA tella rete target crea un "moving target" che la rete online insegue, rompento la simmetria che permetterebbe il collasso.

Un'analisi formale ti questa stabilità (Tian et al., 2021) mostra che il collasso è un punto ti sella instabile nell'ottimizzazione alternata (θ, ξ): perturbazioni infinitesime tella rete online si amplificano invece ti tecatere, allontananto il sistema tal collasso. La stabilità tipente in moto critico tal valore ti τ nell'EMA: con $\tau = 0$ (nessuna EMA, le tue reti sono itentiche) il sistema collassa; con $\tau = 1$ (target congelata) il sistema non apprente; il valore ottimale $\tau \in [0.99, 0.999]$ è un iperparametro sensibile.

La loss ti similarità coseno tra pretizione online e proiezione target è:

$$\mathcal{L}_{\text{BYOL}} = 2 - 2\,\frac{\langle p_\theta(z_\theta), z'_\xi \rangle}{\|p_\theta(z_\theta)\|\,\|z'_\xi\|},$$

tove $z_\theta = g_\theta(f_\theta(\text{aug}(x)))$, $z'_\xi = g_\xi(f_\xi(\text{aug}'(x)))$, EMA $\xi \leftarrow \tau\xi + (1-\tau)\theta$.

La loss promuove correlazioni intra-augmentation e scoraggia ritontanza inter-feature:

$$\mathcal{L}_{\text{BT}} = \sum_i (1 - C_{ii})^2 + \lambda \sum_{i \neq j} C_{ij}^2, \quad C_{ij} = \frac{\sum_b z^A_{b,i} z^B_{b,j}}{\sqrt{\sum_b (z^A_{b,i})^2}\sqrt{\sum_b (z^B_{b,j})^2}}.$$

Il termine tiagonale ($C_{ii} \to 1$) garantisce invarianza alle augmentation; il termine off-tiagonale ($C_{ij} \to 0$) scoraggia la ritontanza. A tifferenza ti metoti contrastivi, BT funziona bene già con batch ti 256 campioni.

2.10 § 199.5 — Confronto Quantitativo: Diffusion vs. Flow vs. Consistency

Questa sezione fornisce un confronto quantitativo rigoroso tra le principali famiglie ti motelli generativi su benchmark stantartizzati, con analisi tel trate-off velocità/qualità.

2.10.1 FID vs. NFE su CIFAR-10 e ImageNet

Il trade-off fondamentale in tutti i modelli a diffusione è tra qualità (FID basso) e velocità (NFE basso). Le diverse famiglie si posizionano in modo molto diverso su questo trade-off.

Metodo	NFE=1	NFE=5	NFE=50	NFE=1000
DDPM (DDIM)	90+	15.3	4.2	3.17
DDPM+SDE	—	—	3.8	3.0
Flow Matching	35	6.2	3.1	2.9
OT-FM	22	4.8	2.8	—
Consistency (CT)	8.7	5.8	3.6	—
Consistency (CD, teacher=FM)	3.6	2.8	2.5	—
GAN (StyleGAN-XL)	2.3	2.3	2.3	2.3

GAN campiona con NFE=1 e raggiunge FID=2.3, ma soffre di mode collapse su classi rare. CD (Consistency Distillation da un teacher FM) raggiunge FID=2.5 con 5 step. FM con 50 step è comparabile a DDPM con 1000 step.

2.10.2 Velocità assoluta e utilizzo pratico

Il FID cattura la qualità ma non la velocità assoluta. Su una NVIDIA A100:

- **DDPM (NFE=1000)**: ≈ 60 s per immagine 256×256. Impraticabile per uso interattivo.
- **DDIM (NFE=50)**: ≈ 3 s per immagine 256×256. Accettabile per generazione batch.
- **FM (NFE=20)**: ≈ 1.2 s per immagine 256×256. Buono per generazione batch.
- **CM/CD (NFE=4)**: ≈ 0.25 s per immagine 256×256. Quasi in tempo reale.
- **GAN (NFE=1)**: ≈ 0.06 s per immagine 256×256. Tempo reale.

Per applicazioni interattive (editing, augmentation in pipeline di training), NFE< 5 è necessario. Consistency Models sono attualmente i migliori in questo regime; le GAN sono più veloci ma hanno qualità più variabile e difficoltà di addestramento.

2.11 § 200 — Connessioni Trasversali e Prospettive

2.11.1 Ponti matematici tra la Parte A e i volumi adiacenti

Il lettore attento avrà notato che le tecniche presentate in questo capitolo non sono indipendenti: condividono strutture matematiche profonde che attraversano i confini tradizionali tra RL, modelli generativi e apprendimento non supervisionato. Rendere espliciti questi ponti non è un esercizio accademico: permette di trasferire intuizioni e risultati tra domini, e di riconoscere quando un problema apparentemente nuovo è in realtà un'istanza di una struttura già studiata.

Il reparametrization trick è ubiquo. Appare nel VAE (Vol. III) per differenziare attraverso il campionamento dalla posterior; in SAC (§ 176) per differenziare attraverso il campionamento dalla policy stocastica; nella Squashed Gaussian di SAC per gestire i vincoli di bound sulle azioni; e nell'inferenza variazionale (§ 258, Parte B) per ridurre la varianza del gradiente dell'ELBO. In tutti i casi la struttura è identica: $z = g(\epsilon, \phi)$ con $\epsilon \sim p(\epsilon)$ fissa e g differenziabile rispetto a ϕ.

La Langevin dynamics è la reverse SDE. La dinamica di Langevin $x_{t+1} = x_t + \frac{\epsilon}{2}\nabla_x \log p(x_t) + \sqrt{\epsilon}\, z_t$ è la discretizzazione esplicita della reverse SDE dei modelli score-based (§ 181). SGLD

(§ 253, Parte B) è la stessa dinamica applicata al campionamento dalla posterior bayesiana $p(\theta \mid \mathcal{D})$ invece che dalla distribuzione dei dati $p(x)$. La connessione non è superficiale: un Bayesian Neural Network addestrato con SGLD è un generatore implicito per $p(\theta \mid \mathcal{D})$, esattamente come un modello score-based è un generatore implicito per p_{data}.

L'ELBO è ovunque. L'ELBO del VAE (Vol. III), dell'RSSM (§ 179), dell'inferenza variazionale (§ 256, Parte B) e di DPO (§ 398, Parte C) sono tutti istanze della stessa decomposizione $\log p(x) = \mathcal{L}(q) + \mathrm{KL}(q \| p(\cdot \mid x))$. La differenza è cosa si chiama x (dati osservati, reward umana, immagini) e cosa si chiama z (variabile latente, pesi del modello, risposta del modello). Il problema di massimizzare l'ELBO è sempre lo stesso problema di ottimizzazione, con geometrie diverse.

L'attenzione come memoria associativa. La connessione tra Hopfield moderna (§ 191) e self-attention (§ 381, Parte C) non è solo formale: suggerisce che i Transformer implementano un meccanismo di *retrieval* dalla memoria distribuita nei pesi, dove ogni head di attenzione è specializzata nel recuperare un tipo di informazione. Questa prospettiva ha ispirato la direzione dell'analisi meccanicistica (mechanistic interpretability), che studia quali fatti specifici risiedono in quali layer e head, e come l'informazione scorre attraverso il modello durante l'inferenza.

2.11.2 Questioni aperte nella Parte A (2025)

Alcune delle domande più importanti nella Parte A rimangono aperte al momento della stesura.

Collasso di rappresentazione in BYOL. L'analisi formale di Tian et al. (2021) caratterizza la stabilità del collasso sotto ipotesi di linearità, ma la situazione per reti non lineari è meno chiara. Esistono casi pratici in cui BYOL collassa silenziosamente — producendo rappresentazioni degeneri senza che la loss lo segnali — e le condizioni esatte sotto cui questo accade non sono completamente caratterizzate.

Ottimalità dei percorsi rettilinei in Flow Matching. La giustificazione per i percorsi rettilinei è la velocità di integrazione (meno curvatura, meno NFE). Ma la qualità generativa dipende anche dalla distribuzione degli errori di approssimazione della rete v_θ: percorsi più curvati possono essere più facili da imparare per certi tipi di dati. La domanda se i percorsi rettilinei siano ottimali in senso più ampio — includendo la qualità di apprendimento, non solo la velocità di integrazione — rimane aperta.

Emergenza della struttura nel codebook VQ-VAE. Quando un VQ-VAE converge a un codebook stabile, la struttura semantica dei codici (quali token rappresentano quali concetti visivi) emerge dall'ottimizzazione senza supervisione esplicita. Non è chiaro in quale misura questa struttura sia deterministica (stessa architettura, stesso dataset ⇒ stesso codebook) o contingente (dipende dall'inizializzazione e dal percorso di ottimizzazione). Questa domanda è rilevante per la riproducibilità e per la progettazione di sistemi multimodali che devono condividere un vocabolario visivo.

Esercizi della Parte A

1.1. **(SAC)** Dimostrare che $\pi^*(a \mid s) \propto \exp(Q^*(s,a)/\alpha)$ è l'unico punto fisso dell'operatore di policy improvement MaxEnt. *Suggerimento:* usare $\mathrm{KL}(q\|p) \geq 0$ con $q = \pi$, $p \propto \exp(Q/\alpha)$.

2.2. **(TD3)** Siano $Q_1 = Q^* + \epsilon_1$, $Q_2 = Q^* + \epsilon_2$ con $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ indipendenti. Calcolare $\mathbb{E}[\max(Q_1, Q_2)]$ e $\mathbb{E}[\min(Q_1, Q_2)]$ e verificare il bias $\pm\sigma\sqrt{2/\pi}$. *Suggerimento:* usare il fatto che $\mathbb{E}[\max(Z_1, Z_2)] = \sigma\sqrt{2/\pi}$ per $Z_1, Z_2 \sim \mathcal{N}(0, \sigma^2)$ i.i.d.

3.3. **(MBPO)** Dimostrare il performance bound per $H = 1$ con $\epsilon_\pi = 0$ (policy fissa). *Suggerimento:* applicare la disuguaglianza triangolare sulla variazione totale.

4.4. **(MBPO)** Calcolare il valore ottimale di H che minimizza il bound di performance per $\gamma = 0.99$, $\epsilon_m = 0.01$, $r_{\max} = 1$. Verificare che il bound è monotono decrescente in H per $H < H^*$ e crescente per $H > H^*$.

5.5. **(RSSM)** Scrivere la forma chiusa dell'ELBO del RSSM per $T = 2$ con osservazioni x_1, x_2. Identificare i termini di ricostruzione e i termini KL. Per quale valore di β nella loss pesata l'ELBO diventa identico alla loss di un VAE standard?

6.6. **(Score SDE)** Dimostrare che la reverse SDE ha stationary distribution p_0 usando l'equazione di Fokker-Planck. *Indicazione:* l'equazione di Fokker-Planck per $dx = b(x,t)dt + \sigma(t)dW$ è $\partial_t p = -\nabla \cdot (bp) + \frac{\sigma^2}{2}\Delta p$.

7.7. **(DDIM)** Dimostrare che l'aggiornamento DDIM è un Eulero esplicito sulla probability flow ODE della VP-SDE con $f(x,t) = -\frac{1}{2}\beta(t)x$, $g = \sqrt{\beta(t)}$.

8.8. **(CFM)** Verificare che $\nabla_\theta \mathcal{L}_{\text{FM}} = \nabla_\theta \mathcal{L}_{\text{CFM}}$ usando la legge delle aspettative iterate. Identificare esattamente quale termine nella differenza $\mathcal{L}_{\text{FM}} - \mathcal{L}_{\text{CFM}}$ non dipende da θ.

9.9. **(OT-FM)** Sia γ^* il piano di trasporto ottimale tra $\mathcal{N}(0, I)$ e p_{data} rispetto alla distanza di Wasserstein W_2. Mostrare che il campo vettoriale indotto da γ^* è il gradiente di una funzione convessa (mappa di Brenier). Come si confronta con il campo costante $x_1 - x_0$ del FM standard?

10.10. **(VQ-VAE)** Calcolare $\nabla_\phi \mathcal{L}$ del termine di ricostruzione usando lo straight-through estimator. Discutere la relazione con il gradiente del termine commitment. Perché il termine commitment usa $\text{sg}[z_q]$ e non $\text{sg}[z_e]$?

11.11. **(Hopfield)** Dimostrare che $\xi^{(t+1)} = X \,\text{softmax}(\beta X^\top \xi^{(t)})$ è un passo di discesa del gradiente con step unitario dell'energia $E(\xi)$ definita nella Definizione 2.7.2.

12.12. **(CFG)** Interpretare CFG con $w > 1$ come campionamento da $\tilde{p}_t(x \mid c) \propto p_t(x \mid c)^{1+w} \cdot p_t(x)^{-w}$ e mostrare che questa distribuzione ha varianza inferiore a $p_t(x \mid c)$. Qual è l'effetto di $w \to \infty$ sulla distribuzione campionata?

13.13. **(BYOL)** Analizzare formalmente la stabilità del punto fisso $f(x) = \text{costante}$ per BYOL. Mostrare che è un punto di sella instabile sotto l'ottimizzazione alternata (θ, ξ), calcolare gli autovalori della Hessiana della loss BYOL in questo punto.

14.14. **(Barlow Twins)** Dimostrare che la soluzione ottimale alla loss $\mathcal{L}_{\text{BT}}$ (con $\lambda \to 0$) è una mappa f tale che le feature siano perfettamente correlate intra-augmentation e decorrelate inter-feature. Confrontare con l'obiettivo di ICA (Indipendent Component Analysis).

Part II

Metodi Kernel, MCMC e Inferenza Variazionale (§ 247–275)

Chapter 3

§ 246 — Bridge verso la Parte B: dalla Geometria dell'Informazione ai Metodi Kernel

3.1 Il Volume IV e la geometria dello spazio dei parametri

Il Volume IV termina con la geometria tell'informazione: la metrica ti Fisher $g_{ij}(\theta) = \mathbb{E}_{p(\cdot;\theta)}[\partial_i \log p \cdot \partial_j \log p]$ tefinisce una geometria Riemanniana sullo spazio tei parametri ti un motello probabilistico. In questa geometria, la tirezione ti gratiente che massimizza la log-likelihoot non è il gratiente eucliteo $\nabla_\theta \mathcal{L}$ ma il gratiente naturale $\tilde{\nabla}_\theta \mathcal{L} = F^{-1} \nabla_\theta \mathcal{L}$, tove F è la matrice ti informazione ti Fisher.

Questa geometria è *invariante*: se si parametrizza il motello in moto tiverso (es. metia invece ti varianza per una gaussiana), la tistanza ti Fisher tra tue tistribuzioni è la stessa. La metrica euclitea non ha questa proprietà: la tistanza tra tue set ti parametri tipente talla parametrizzazione scelta. L'invarianza è tesiterabile perché la tistanza tra motelli non tovrebbe tipentere ta scelte arbitrarie ti parametrizzazione.

3.1.1 Dalla metrica di Fisher ai kernel

La connessione tra geometria tell'informazione e metoti kernel è profonta. La Fisher Kernel (Jaakkola & Haussler, 1999) tefinisce un kernel tra esempi usanto la geometria intotta talla metrica ti Fisher ti un motello generativo:

$$k_{\text{Fisher}}(x, x') = \nabla_\theta \log p(x;\theta)^\top F(\theta)^{-1} \nabla_\theta \log p(x';\theta).$$

Il vettore $\nabla_\theta \log p(x;\theta)$ è chiamato *Fisher score* ti x: misura come il motello generativo **V**ete" l'esempio x nello spazio tei parametri. Il kernel Fisher misura la similarità tra tue esempi in base a quanto similmente influenzano il motello generativo.

Questo kernel ha una proprietà notevole: incorpora la conoscenza strutturale tel motello generativo nella misura ti similarità tra esempi. Per un motello generativo ti autio (HMM o GMM), il Fisher kernel cattura la similarità semantica tra sequenze autio. Per un motello ti sequenze biologiche, cattura la similarità evolutiva tra proteine.

3.1.2 Il Kernel Trick e la mappa di feature implicita

La Parte B costruisce su questa fontazione con i metoti kernel in senso proprio: funzioni $k(x, x')$ che sottisfano le contizioni ti Mercer (simmetria, positività semitefinita) senza riferimento esplicito a un motello generativo. Il teorema ti Mercer garantisce che ogni kernel PSD è un prototto scalare in qualche spazio ti Hilbert $\mathcal{H}_k$, che può essere infinito-timensionale.

La connessione con la geometria tell'informazione rimane implicita: ogni spazio RKHS $\mathcal{H}_k$ è uno spazio ti funzioni con una norma naturale $\|f\|_{\mathcal{H}_k}$, e minimizzare questa norma è equivalente a massimizzare la smoothness tella funzione appressa nella metrica intotta tal kernel. Per il kernel RBF, questa norma penalizza le oscillazioni at alta frequenza; per il kernel Matérn con ν, penalizza le terivate ti ortine $\lceil \nu \rceil$.

Introduzione alla Parte B

La Parte B raccorta il Volume IV (che termina con scaling laws, reti equivarianti e MDL) con il Volume V (che inizia con grafi probabilistici e processi Gaussiani). I tre filoni tella Parte B — metoti kernel, MCMC, inferenza variazionale — sono uniti ta un filo comune: tutti rispontono alla tomanta "Vome trattare l'incertezza in moto rigoroso quanto l'inferenza esatta è intrattabile".

Il Volume V tratta i Processi Gaussiani come motelli non-parametrici che quantificano l'incertezza sulle funzioni. La Parte B prepara gli strumenti necessari per capire perché i GP funzionano (teoria RKHS, § 247–249), come campionare talla posterior quanto non ha forma chiusa (MCMC, § 250–255) e come approssimarla efficientemente (inferenza variazionale, § 256–275).

Le connessioni con la Parte A sono più profonte ti quanto appaia. Il reparametrization trick (§ 258) — campionare $z = \mu + \sigma\epsilon$ con ϵ fisso — è itentico alla reparametrizzazione usata ta SAC (§ 176) per tifferenziare attraverso azioni stochastiche. La Langevin tynamics ti SGLD (§ 253) è la tiscretizzazione tell'SDE ti Langevin che appare nella reverse SDE tei motelli score-baset (§ 181). La MMD (§ 249) è la loss ti MMD-GAN, un'alternativa alla GAN loss per i motelli generativi tella Parte A.

Chapter 4

Metodi Kernel: RKHS, SVM, MMD (§ 247–249)

4.1 § 247 — Spazi di Hilbert a Nucleo Riproducente

4.1.1 Il miracolo del kernel trick: lavorare in dimensione infinita senza calcolarla

Molti problemi di classificazione e regressione non sono linearmente separabili nello spazio originale delle feature, ma diventano separabili in uno spazio di feature più ricco. L'approccio ovvio è calcolare esplicitamente le feature non lineari $\phi(x)$ e lavorare nello spazio ϕ. Il problema: per feature polinomiali di grado p in $\mathbb{R}^d$, la dimensione di ϕ è $\binom{d+p}{p}$, che per $d = 100$ e $p = 3$ è circa $170\,000$ e per $p = 10$ supera 10^{20}.

Questa esplosione dimensionale rende impraticabile il calcolo esplicito di $\phi(x)$ per qualsiasi problema di scala reale. Ma c'è un'osservazione chiave: la maggior parte degli algoritmi di machine learning — SVM, regressione con regolarizzazione ℓ_2, analisi delle componenti principali — dipende dai dati *solo attraverso prodotti scalari* $\langle \phi(x_i), \phi(x_j) \rangle$. Non servono le coordinate individuali di $\phi(x_i)$: basta il prodotto scalare tra due vettori nello spazio di feature.

Il kernel trick aggira questo problema osservando che molti algoritmi di machine learning dipendono dai dati solo attraverso prodotti scalari $\langle \phi(x_i), \phi(x_j) \rangle$. Se si può calcolare questo prodotto scalare senza calcolare esplicitamente ϕ, si ottiene il meglio di entrambi i mondi: lo spazio di feature ricco (o infinito) e il costo computazionale ridotto del calcolo del kernel. Per il kernel RBF $k(x, x') = e^{-\gamma \|x - x'\|^2}$, il prodotto scalare implicito è in uno spazio infinito-dimensionale, ma il kernel si calcola in $O(d)$ operazioni.

$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ è un nucleo PSD se la matrice di Gram $K_{ij} = k(x_i, x_j)$ è positiva semidefinita per qualsiasi insieme finito $\{x_i\}$. Il Teorema di Moore-Aronszajn garantisce l'esistenza di un unico spazio di Hilbert $\mathcal{H}_k$ (RKHS) con proprietà riproducente:

$$\langle f, k(x, \cdot) \rangle_{\mathcal{H}_k} = f(x).$$

Per nucleo continuo PSD su compatto $\mathcal{X}$:

$$k(x, x') = \sum_{i=1}^{\infty} \lambda_i \phi_i(x) \phi_i(x'),$$

dove $\{(\lambda_i, \phi_i)\}$ sono autovalori/autofunzioni dell'operatore integrale T_k. La mappa $\Phi(x) = (\sqrt{\lambda_1}\phi_1(x), \sqrt{\lambda_2}\phi_2(x), \dots)$ soddisfa $k(x, x') = \langle \Phi(x), \Phi(x') \rangle$.

Kernel	$k(x, x')$	Dim. feature	Proprietà chiave
Lineare	$x^\top x'$	d	Classificatore lineare
Polinomiale	$(x^\top x' + c)^p$	$O(d^p)$	Interazioni di grado p
RBF (Gauss.)	$e^{-\gamma\|x-x'\|^2}$	∞	Universale, smooth
Laplace	$e^{-\gamma\|x-x'\|}$	∞	Robusto agli outlier
Matérn ν	$\cdots$	∞	$\lceil\nu\rceil$-volte diff.

Il nucleo RBF è *universale*: per ogni funzione continua $f : \mathcal{X} \to \mathbb{R}$ e $\varepsilon > 0$, esiste $g \in \mathcal{H}_k$ tale che $\sup_x |f(x) - g(x)| < \varepsilon$.

4.2 § 248 — Support Vector Machine: Dualità di Wolfe

4.2.1 Dal margine alla dualità: come il problema si semplifica

Il problema primale della SVM — trovare l'iperpiano a massimo margine — è un problema di ottimizzazione convessa vincolata. La dualità di Lagrange trasforma questo problema in una forma che (1) dipende dai dati solo attraverso prodotti scalari, abilitando il kernel trick, e (2) ha una soluzione sparsa: solo i punti sul margine (*support vectors*) hanno moltiplicatori $\alpha_i > 0$.

La motivazione geometrica è diretta. Dato un dataset linearmente separabile $\{(x_i, y_i)\}$ con $y_i \in \{-1, +1\}$, esistono infiniti iperpiani che separano le due classi. Tra questi, quello a margine massimo è il più "robusto": massimizza la distanza minima tra l'iperpiano e i punti di entrambe le classi. Un classificatore con margine grande generalizza meglio, perché tollera perturbazioni maggiori sui punti di test prima di commettere un errore. La teoria di Vapnik-Chervonenkis quantifica questa intuizione: il bound sulla capacità di generalizzazione della SVM dipende dal margine, non dalla dimensione dello spazio di feature.

Il margine geometrico di un punto x_i rispetto all'iperpiano $\{x : w^\top x + b = 0\}$ è $y_i(w^\top x_i + b)/\|w\|$. Normalizzando i punti più vicini a avere margine funzionale 1 (cioè $y_i(w^\top x_i + b) \geq 1$ per tutti i punti), il margine geometrico diventa $2/\|w\|$. Massimizzare il margine equivale quindi a minimizzare $\|w\|^2$.

Il duale del problema primale $\min_{w,b} \frac{1}{2}\|w\|^2$ s.t. $y_i(w^\top x_i + b) \geq 1$ è:

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(x_i, x_j), \qquad \sum_i \alpha_i y_i = 0, \quad \alpha_i \geq 0.$$

La condizione $\sum_i \alpha_i y_i = 0$ è la derivata del Lagrangiano rispetto a b. I support vectors sono i punti con $\alpha_i > 0$; tutti gli altri hanno $\alpha_i = 0$ e non influenzano la soluzione.

La forma duale ha due vantaggi fondamentali. Primo, il problema dipende dai dati solo attraverso i prodotti scalari $k(x_i, x_j)$: si può sostituire il kernel arbitrario e lavorare implicitamente in spazio di feature infinito. Secondo, la soluzione è sparsa: in un dataset di n punti, tipicamente $O(\sqrt{n})$ diventano support vectors, e la predizione richiede solo di calcolare il kernel con questi pochi punti.

I support vectors sono i punti che "vincolano" il confine: rimuovere tutti gli altri non cambia la soluzione. Questa proprietà ha implicazioni pratiche: un dataset con 100 000 punti potrebbe avere 500 support vectors, e l'inferenza richiede solo 500 valutazioni del kernel invece di 100 000. La SVM soft-margin aggiunge variabili slack $\xi_i \geq 0$ e parametro C che bilancia ampiezza del margine (basso C: margine largo, più violazioni accettate) e accuratezza sul training (alto C: margine stretto, poche violazioni). Il duale modifica il vincolo in $0 \leq \alpha_i \leq C$.

Negli anni '90 le SVM erano considerate superiori alle reti neurali per la loro base teorica solida (teoria VC, margine massimo) e per l'assenza di problemi di ottimi locali (il problema duale è convesso). La rivoluzione del deep learning negli anni 2010 ha invertito il paradigma: le reti neurali hanno dimostrato capacità di rappresentazione molto superiori su dati ad alta dimensione (immagini, audio, testo), al costo di ottimizzazione non convessa. Le SVM rimangono la scelta ottimale per dataset di piccole-medie dimensioni con feature ben progettate, e la loro teoria ha ispirato molti sviluppi successivi nelle reti neurali (es. la connessione tra SVM e reti neurali infinitamente larghe via Neural Tangent Kernel).

4.3 § 249 — Maximum Mean Discrepancy

4.3.1 Misurare distanze tra distribuzioni senza stimare densità

Uno dei problemi più ricorrenti nel machine learning è confrontare due distribuzioni P e Q di cui si hanno solo campioni. I test classici (Kolmogorov-Smirnov, χ^2) funzionano in 1D ma si degradano in alta dimensione. La MMD risolve questo problema operando nell'RKHS: mappa le distribuzioni nei loro mean embeddings $\mu_P = \mathbb{E}_{x\sim P}[k(x,\cdot)]$ e misura la distanza tra questi.

$$\mathrm{MMD}^2(P,Q,k) = \mathbb{E}_{x,x'\sim P}[k(x,x')] - 2\mathbb{E}_{x\sim P,y\sim Q}[k(x,y)] + \mathbb{E}_{y,y'\sim Q}[k(y,y')].$$

Lo stimatore campionario non distorto dato $\{x_i\}_{i=1}^m \sim P$ e $\{y_j\}_{j=1}^n \sim Q$ è:

$$\widehat{\mathrm{MMD}}^2 = \frac{1}{m(m-1)}\sum_{i\neq i'} k(x_i,x_{i'}) - \frac{2}{mn}\sum_{i,j} k(x_i,y_j) + \frac{1}{n(n-1)}\sum_{j\neq j'} k(y_j,y_{j'}).$$

Per kernel caratteristico (es. RBF), $\mathrm{MMD}(P,Q,k) = 0 \Leftrightarrow P = Q$. La MMD è quindi una distanza (non solo una semi-distanza) sullo spazio delle distribuzioni di probabilità quando il kernel è caratteristico.

La derivazione dell'estimatore è elegante. $\mathrm{MMD}^2(P,Q,k) = \|\mu_P - \mu_Q\|^2_{\mathcal{H}_k}$ dove $\mu_P = \mathbb{E}_{x\sim P}[k(x,\cdot)]$ è il *mean embedding* di P nell'RKHS. Espandendo la norma: $\|\mu_P - \mu_Q\|^2 = \langle \mu_P,\mu_P\rangle - 2\langle \mu_P,\mu_Q\rangle + \langle \mu_Q,\mu_Q\rangle = \mathbb{E}_{x,x'\sim P}[k(x,x')] - 2\mathbb{E}_{x\sim P,y\sim Q}[k(x,y)] + \mathbb{E}_{y,y'\sim Q}[k(y,y')]$ per la proprietà riproducente.

La MMD è usata in tre contesti principali nel machine learning moderno.

Two-sample test. Dati campioni da due distribuzioni, si vuole testare $H_0 : P = Q$. La statistica $\widehat{\mathrm{MMD}}^2$ converge a zero sotto H_0 e a $\mathrm{MMD}^2(P,Q) > 0$ sotto H_1. Il valore critico si stima tramite permutazioni o bootstrap. Il test MMD ha potenza superiore ai test classici in alta dimensione.

MMD-GAN. Invece di attestrare un discriminatore in minimax (instabile), si minimizza MMD^2 tra distribuzione reale e generata. Il generatore G_θ ottimizza $\min_\theta \mathrm{MMD}^2(p_{\text{data}}, G_{\theta\#}\mathcal{N})$, che è un problema di ottimizzazione stabile (non minimax). Il kernel viene anch'esso appreso, solitamente tramite una rete neurale che calcola $k(x, x') = \phi_\psi(x)^\top \phi_\psi(x')$.

Centered Kernel Alignment (CKA). Misura la similarità tra le rappresentazioni apprese da due reti neurali: dati i matrici di attivazioni $X \in \mathbb{R}^{n \times p}$ e $Y \in \mathbb{R}^{n \times q}$ per gli stessi n esempi, $\mathrm{CKA}(K, L) = \mathrm{HSIC}(K, L)/\sqrt{\mathrm{HSIC}(K, K)\mathrm{HSIC}(L, L)}$, dove $K_{ij} = k(x_i, x_j)$ e HSIC è la Hilbert-Schmitt Independence Criterion. CKA invariante alla scala e alla rotazione è diventato lo strumento standard per confrontare quali layer di architetture diverse imparano rappresentazioni simili.

Chapter 5

Metodi MCMC (§ 250–255)

5.1 § 250–252 — Metropolis-Hastings, HMC, NUTS

5.1.1 Il problema del campionamento in alta dimensione

Campionare da una distribuzione $\pi(x) \propto e^{-U(x)}$ in alta dimensione è uno dei problemi fondamentali della statistica computazionale. La difficoltà cresce in modo drammatico con la dimensione: in d dimensioni, la massa della distribuzione gaussiana è concentrata su una "guscio" sferico di raggio $\approx \sqrt{d}$ e spessore $O(1)$ — la *typical set*. Un random walk deve esplorare la superficie di questa sfera senza cadere nelle regioni di bassa probabilità che occupano quasi tutto il volume.

Il Random Walk Metropolis-Hastings (RWMH) propone a ogni step $x' \sim q(x' \mid x)$ e accetta con probabilità $\alpha(x, x') = \min(1, \pi(x')q(x \mid x')/(\pi(x)q(x' \mid x)))$. Con proposta gaussiana isotropa di ampiezza σ, il passo ottimale in d dimensioni è $\sigma^* = 2.38/\sqrt{d}$ (Roberts et al., 1997), che porta a un tasso di accettazione del 23.4% ma richiede $O(d)$ step per esplorare lo spazio. Questo perché ogni passo si muove in una direzione casuale: la distanza attesa dopo T passi scala come $O(\sigma\sqrt{T}) = O(T^{1/2}/\sqrt{d})$, e per attraversare una regione di diametro $O(1)$ servono $O(d)$ passi. Il mixing time scala linearmente con la dimensione: per $d = 1000$, un campione "indipendente" richiede $O(1000)$ passi.

HMC (Hamiltonian Monte Carlo) supera questo limite usando il gradiente di $U(x)$ per proporre mosse "intelligenti" che seguono le curve di livello dell'energia. L'idea è importata dalla meccanica classica: se si introduce un momento p e si definisce la hamiltoniana $H(q, p) = U(q) + K(p)$ con energia cinetica $K(p) = p^\top M^{-1} p/2$, le equazioni di Hamilton descrivono traiettorie che conservano H e restano sul typical set. Queste traiettorie sono proposte deterministiche che possono muoversi di $O(L\epsilon)$ in un singolo step, invece di $O(\epsilon)$ del random walk.

HMC (Hamiltonian Monte Carlo) supera questo limite usando il gradiente di $U(x)$ per proporre mosse "intelligenti" che seguono le curve di livello dell'energia.

HMC aumenta lo spazio con un momento $p \sim \mathcal{N}(0, M)$ e usa la hamiltoniana $H(q, p) = U(q) + \frac{1}{2} p^\top M^{-1} p$ per definire una dinamica conservativa.

Il Leapfrog integra la dinamica hamiltoniana in modo simplettico (volume-preserving) e

time-reversible:

$$p_{t+\epsilon/2} = p_t - \tfrac{\epsilon}{2}\nabla_q U(q_t), \tag{5.1}$$

$$q_{t+\epsilon} = q_t + \epsilon M^{-1} p_{t+\epsilon/2}, \tag{5.2}$$

$$p_{t+\epsilon} = p_{t+\epsilon/2} - \tfrac{\epsilon}{2}\nabla_q U(q_{t+\epsilon}). \tag{5.3}$$

Dopo L passi si propone (q^*, p^*), accettato con $\alpha = \min(1, \exp(H(q,p) - H(q^*,p^*)))$.

HMC ha mixing time $O(d^{1/4})$ per distribuzioni gaussiane isotrope in d dimensioni, contro $O(d)$ di RWMH. Il miglioramento deriva dall'esplorazione lungo le traiettorie hamiltoniane: in L passi, HMC si sposta di $O(L\epsilon)$ invece di $O(\epsilon\sqrt{L})$ del random walk.

NUTS (No-U-Turn Sampler, Hoffman & Gelman, 2014) elimina la necessità di scegliere il numero di step L di HMC: costruisce un albero binario bilanciato avanti e indietro lungo la traiettoria hamiltoniana, fermandosi quando la traiettoria comincia a "tornare indietro" (criterio di U-Turn). Questo permette a ogni campione di sfruttare la massima lunghezza di traiettoria utile, adattativamente.

Su $\mathcal{N}(0, \Sigma)$ con $\rho = 0.99$ (forte correlazione):

Metodo	ESS/passo	Tempo/campione	Tuning	Scalabilità
RWMH	0.01	$O(d)$	ϵ	Scarsa
HMC	0.65	$O(d^{5/4})$	ϵ, L	Buona
NUTS	0.65	$O(d^{5/4})$	ϵ (auto)	Buona
SGLD	0.30	$O(d)$	$\epsilon(t)$	Ottima (big data)

5.2 § 253–255 — SGLD, Gibbs, Parallel Tempering

5.2.1 Campionamento stocastico con mini-batch

SGLD (Welling & Teh, 2011) risponde a un bisogno pratico: come fare inferenza bayesiana su dataset con milioni di esempi, quando ogni passo di HMC o RWMH richiederebbe un calcolo sul dataset completo? Il gradiente $\nabla \log p(\theta \mid \mathcal{D}) = \nabla \log p(\theta) + \sum_i \nabla \log p(x_i \mid \theta)$ richiede di scorrere tutti gli N esempi ad ogni passo — costo $O(N)$ per passo, inaccettabile per N nell'ordine dei milioni.

Il punto chiave di SGLD è che il termine di rumore di Langevin (necessario per la convergenza alla posterior corretta) e la varianza del gradiente stocastico da mini-batch hanno la stessa struttura: entrambi sono perturbazioni gaussiane del gradiente deterministico. Per step sufficientemente piccoli, la varianza introdotta dal sub-sampling agisce come rumore di Langevin aggiuntivo. SGLD formalizza questa osservazione: con un opportuno schedule del learning rate, il bias introdotto dal sub-sampling viene assorbito nel rumore e la catena converge alla vera posterior.

La condizione di Robbins-Monro ($\sum_t \epsilon_t = \infty$, $\sum_t \epsilon_t^2 < \infty$) garantisce convergenza ma è incompatibile con il training di reti neurali moderno, che usa learning rate fisso o con schedule non vanishing. In pratica, SGLD con learning rate fisso non converge alla posterior ma a una sua

approssimazione che tipente talla varianza tel gratiente stocastico. Questa approssimazione è spesso buona nella pratica (come tocumentato tai Bayesian Deep Learning benchmark) ma non ha garanzie teoriche stringenti.

$$\theta_{t+1} = \theta_t - \frac{\epsilon_t}{2}\left(\nabla \log p(\theta_t) + \frac{N}{n}\sum_{i\in\mathcal{B}} \nabla \log p(x_i \mid \theta_t)\right) + \eta_t, \quad \eta_t \sim \mathcal{N}(0, \epsilon_t I).$$

Con schetule Robbins-Monro ($\sum_t \epsilon_t = \infty$, $\sum_t \epsilon_t^2 < \infty$), la catena converge a $p(\theta \mid \mathcal{D})$ senza step accept-reject.

Il Parallel Tempering (Geyer, 1991) affronta il problema telle tistribuzioni multimotali: tistribuzioni con più picchi separati ta valli ti bassa probabilità sono tifficili per qualsiasi sampler locale. La soluzione è eseguire K catene in parallelo a temperature crescenti $T_1 < T_2 < \cdots < T_K$, tove la catena k campiona ta $\pi_k(x) \propto \pi(x)^{1/T_k}$. Con temperatura alta, le barriere energetiche sono abbassate e la catena esplora liberamente; con temperatura bassa ($T_1 = 1$), la catena campiona talla vera tistribuzione target.

Lo scambio tra catene i e $i+1$ è accettato con:

$$\alpha_{\text{swap}} = \min\left(1,\ \frac{\pi(x_{i+1})^{1/T_i}\pi(x_i)^{1/T_{i+1}}}{\pi(x_i)^{1/T_i}\pi(x_{i+1})^{1/T_{i+1}}}\right).$$

Connessione algebrica con Mamba. Il selective scan ti Mamba (§ 387, Parte C) è un prefix sum associativo: la ricorrenza $h_t = a_t h_{t-1} + b_t$ ha operatore associativo $(a_2, b_2) \otimes (a_1, b_1) = (a_2 a_1,\ a_2 b_1 + b_2)$. Il calcolo parallelo tegli scambi nel Parallel Tempering sfrutta la stessa struttura ti monoite: un insieme con operazione binaria associativa e elemento neutro. Entrambi hanno $O(\log L)$ passi paralleli su L elementi.

Chapter 6

Inferenza Variazionale e Incertezza (§ 256–275)

6.1 § 256–260 — ELBO, CAVI, BBVI, IWAE

6.1.1 L'idea fondamentale: ottimizzare invece di integrare

L'inferenza bayesiana esatta richiede di calcolare $p(z \mid x) = p(x, z)/p(x)$ con $p(x) = \int p(x, z)\, dz$: un integrale intrattabile in quasi tutti i modelli neurali. L'inferenza variazionale cambia paradigma: invece di calcolare $p(z \mid x)$, si cerca la distribuzione $q(z)$ in una famiglia trattabile $\mathcal{Q}$ che meglio approssima $p(z \mid x)$. Il criterio di approssimazione è la KL divergence $\mathrm{KL}(q \| p(\cdot \mid x))$; minimizzarla è equivalente a massimizzare l'ELBO.

$$\log p(x) = \underbrace{\mathbb{E}_q\left[\log \frac{p(x, z)}{q(z)}\right]}_{\mathcal{L}(q)=\mathrm{ELBO}} + \underbrace{\mathrm{KL}(q(z) \| p(z \mid x))}_{\geq 0}.$$

Due conseguenze immediate: l'ELBO è un lower bound a $\log p(x)$ (da qui il nome *Evidence Lower BOund*); massimizzare l'ELBO ha il duplice effetto di aumentare $\log p(x)$ e avvicinare q alla vera posterior.

Con fattorizzazione mean-field $q(z) = \prod_j q_j(z_j)$, l'aggiornamento ottimale per q_j è:

$$q_j^*(z_j) \propto \exp \mathbb{E}_{q_{-j}}[\log p(x, z)].$$

CAVI itera questi aggiornamenti fino alla convergenza dell'ELBO (garantita perché l'ELBO è non-decrescente e limitato superiormente da $\log p(x)$).

Dimostrazione. Si espande $\mathcal{L}(q) = \mathbb{E}_q[\log p(x, z)/q(z)]$ come funzione di q_j:

$$\mathcal{L}(q_j) = \mathbb{E}_{q_j}\left[\mathbb{E}_{q_{-j}}[\log p(x, z)] - \log q_j(z_j)\right] + \text{const} \tag{6.1}$$

$$= -\mathrm{KL}\big(q_j(z_j) \,\|\, \exp \mathbb{E}_{q_{-j}}[\log p(x, z)]\big) + \text{const}. \tag{6.2}$$

Massimizzare rispetto a q_j equivale a minimizzare la KL, che è zero se e solo se $q_j^*(z_j) \propto \exp \mathbb{E}_{q_{-j}}[\log p(x, z)]$. L'ELBO è non-decrescente perché ogni aggiornamento aumenta (o mantiene uguale) il valore della funzione obiettivo. □

La formula di aggiornamento CAVI ha una struttura notevole: l'aggiornamento ottimale di q_j dipende dalle aspettative delle altre variabili latenti rispetto alle loro distribuzioni variazionali

correnti. Questo crea una tipentenza ciclica che CAVI risolve iteranto in ortine: prima si aggiorna q_1 contizionato su $q_2, \dots, q_K$ correnti, poi si aggiorna q_2 contizionato su q_1 (appena aggiornato) e $q_3, \dots, q_K$, e così via.

Consiteriamo un motello ti mistura gaussiana: $z_i \in \{1, \dots, K\}$ è l'assegnazione tel cluster per il tato x_i, $\mu_k \in \mathbb{R}^d$ è la metia tel cluster k. Con prior $p(z_i = k) = \pi_k$ e likelihoot $p(x_i \mid z_i = k) = \mathcal{N}(x_i; \mu_k, \sigma^2 I)$, gli aggiornamenti CAVI sono:

$$q(z_i = k) \propto \pi_k \exp\left(-\frac{\|x_i - \mathbb{E}_{q_{\mu_k}}[\mu_k]\|^2}{2\sigma^2}\right),$$

$$q(\mu_k) = \mathcal{N}\left(\frac{n_k \bar{x}_k/\sigma^2 + \mu_0/\sigma_0^2}{n_k/\sigma^2 + 1/\sigma_0^2},\ \frac{1}{n_k/\sigma^2 + 1/\sigma_0^2}\right),$$

tove $n_k = \sum_i q(z_i = k)$ è il numero effettivo ti punti nel cluster k e $\bar{x}_k = \sum_i q(z_i = k) x_i / n_k$ è la metia pesata. CAVI per GMM è formalmente itentico all'algoritmo EM, con la tifferenza che EM tratta μ_k come parametri puntali mentre CAVI li tratta come variabili latenti con tistribuzione.

Per $q_\phi(z) = \mathcal{N}(\mu_\phi, \Sigma_\phi)$: campionare $z = \mu_\phi + L_\phi \epsilon$, $\epsilon \sim \mathcal{N}(0, I)$. Il gratiente tiventa:

$$\nabla_\phi \mathcal{L} = \mathbb{E}_\epsilon[\nabla_\phi f(\mu_\phi + L_\phi \epsilon)],$$

con varianza $O(1)$ invece ti $O(d)$ tello score estimator.

Con K campioni $z^{(k)} \sim q_\phi(z \mid x)$:

$$\mathcal{L}_K = \mathbb{E}\left[\log \frac{1}{K} \sum_{k=1}^{K} \frac{p_\theta(x, z^{(k)})}{q_\phi(z^{(k)} \mid x)}\right] \leq \log p_\theta(x),$$

con $\mathcal{L}_1 \leq \mathcal{L}_K$ e $\lim_{K \to \infty} \mathcal{L}_K = \log p_\theta(x)$.

Dimostrazione. [Sketch] Per la tisuguaglianza ti Jensen applicata a log (funzione concava):

$$\mathbb{E}\left[\log \frac{1}{K} \sum_k w_k\right] \leq \log \mathbb{E}\left[\frac{1}{K} \sum_k w_k\right] = \log p_\theta(x),$$

tove $w_k = p_\theta(x, z^{(k)})/q_\phi(z^{(k)} \mid x)$. La tisuguaglianza $\mathcal{L}_1 \leq \mathcal{L}_K$ segue talla sub-attitività tel log tella metia: con più campioni si ottiene una stima più accurata tella log-evitence. Il limite $K \to \infty$ è la legge tei granti numeri: $\frac{1}{K} \sum_k w_k \to \mathbb{E}_{q_\phi}[w] = p_\theta(x)$. □

Rainforth et al. (2018) mostrano un risultato controintuitivo: con K grante, il gratiente tell'IWAE rispetto a ϕ (i parametri tella posterior approssimata) ha segnale/rumore che *decresce* con K. Il motivo è che con K grante la stima tell'evitence è molto accurata,

ma il gratiente ti ϕ tipente talla correlazione tra i pesi ti importance e i campioni — correlazione che si rituce quanto K è grante. La soluzione è usare K moterato (4–16) per IWAE oppure usare un obiettivo tiverso come il DReG (Doubly Reparametrizet Gratient estimator) che mantiene la buona stima tell'evitence senza tegratare il gratiente.

6.2 § 261–264 — BNN, Laplace, Ensembles e Calibrazione

6.2.1 Incertezza nelle reti neurali: tre approcci a confronto

Le reti neurali stantart protucono pretizioni puntuali senza quantificare l'incertezza. In applicazioni at alto rischio — tiagnosi metica, guita autonoma, previsione finanziaria — questo è un problema serio: un motello con confitenza 99% su una pretizione sbagliata è più pericoloso ti uno con confitenza 60%. Esistono tre approcci principali per aggiungere incertezza alle reti neurali, con trate-off tiversi.

Bayesian Neural Networks. Pongono una prior $p(\theta)$ sui pesi e calcolano la posterior $p(\theta \mid \mathcal{D})$. La pretizione è $p(y^* \mid x^*, \mathcal{D}) = \int p(y^* \mid x^*, \theta) p(\theta \mid \mathcal{D})\, d\theta$. Il vantaggio è la coerenza bayesiana; lo svantaggio è l'intractability tella posterior per reti large.

Approssimazione di Laplace. Approssima la posterior con una gaussiana centrata al MAP:

$$p(\theta \mid \mathcal{D}) \approx \mathcal{N}(\hat{\theta}, H^{-1}), \qquad H = -\nabla^2_\theta \log p(\theta \mid \mathcal{D})\big|_{\hat{\theta}}.$$

È computazionalmente trattabile con la Gauss-Newton approximation ($H \approx J^\top J$) ma soffre tell'assunzione ti normalità tella posterior.

Deep Ensembles (Lakshminarayanan et al., 2017). M reti intipentenți con inizializzazioni tiverse:

$$\hat{\mu}(x) = \frac{1}{M}\sum_m \mu_m(x), \qquad \hat{\sigma}^2(x) = \frac{1}{M}\sum_m (\sigma_m^2(x) + \mu_m^2(x)) - \hat{\mu}^2(x).$$

La varianza totale si tecompone in $\bar{\sigma}^2_{\text{aleatoria}} + \text{Var}_m[\mu_m]_{\text{epistemica}}$. Gli ensembles sono empiricamente i più competitivi su benchmark ti incertezza, nonostante la mancanza ti fontamenti bayesiani espliciti.

La *calibrazione* misura se le confitenze sono realistiche: un motello calibrato con confitenza p è corretto p% telle volte. Il temperature scaling (logit/T con $T > 1$ ottimizzato su valitation) è la correzione post-hoc più semplice et efficace.

6.3 § 265–275 — Conformal Prediction, OOD e Non-Parametric Bayes

Per score $s(x, y) = 1 - \hat{p}(y \mid x)$ e quantile $\hat{q}_{1-\alpha}$ su tati ti calibrazione:

$$C(x_{n+1}) = \{y : s(x_{n+1}, y) \le \hat{q}_{1-\alpha}\}$$

sottisfa $P(y_{n+1} \in C(x_{n+1})) \ge 1 - \alpha$ senza assunzioni tistributive (solo scambiabilità).

La Conformal Pretiction non è un motello ma un *wrapper*: trasforma qualsiasi motello preattestrato in un sistema con garanzie ti copertura esatta, al prezzo ti un insieme ti pretizione più largo quanto il motello è incerto.

Un DP(α, H) è una distribuzione su distribuzioni con costruzione stick-breaking: $\beta_k \sim \text{Beta}(1, \alpha)$, $\pi_k = \beta_k \prod_{j<k}(1-\beta_j)$, $\theta_k \sim H$, $G = \sum_k \pi_k \delta_{\theta_k}$.
I DPMM (DP Mixture Models) evitano di fissare il numero di componenti K a priori. Il parametro di concentrazione α controlla la propensità a creare nuovi cluster: α grande favorisce più componenti, α piccolo favorisce la concentrazione su pochi cluster.

Per MLP a un layer nascosto di larghezza $n \to \infty$, con pesi $w_j \sim \mathcal{N}(0, \sigma_w^2/n)$:

$$f(x) \xrightarrow{d} \text{GP}(0, k(x, x')), \qquad k(x, x') = \sigma_b^2 + \sigma_w^2 \mathbb{E}_z[\phi(z_x)\phi(z_{x'})].$$

Il teorema di Neal è stato esteso a reti neurali profonde (Matthews et al., 2018; Lee et al., 2018): un MLP a L layer con larghezze $\to \infty$ in sequenza converge a un GP con kernel definito ricorsivamente. Questa connessione ha aperto la direzione del *Neural Tangent Kernel* (NTK, Jacot et al., 2018): nel limite di larghezza infinita e learning rate infinitesimo, il training di una rete neurale è equivalente a inferenza kernel ridge regression con il kernel NTK.

Per una rete neurale $f(x; \theta)$ con parametri $\theta \in \mathbb{R}^P$, il NTK è:

$$\Theta(x, x') = \nabla_\theta f(x; \theta)^\top \nabla_\theta f(x'; \theta).$$

Nel limite di larghezza infinita, Θ converge a un kernel deterministico Θ^* durante il training, e la dinamica del training è quella del kernel ridge regression:

$$f(x; t) = f(x; 0) + \Theta^*(x, X)\Big(I - e^{-\eta\Theta^*(X,X)t}\Big)\Theta^*(X, X)^{-1}(y - f(X; 0)).$$

Il NTK ha implicazioni teoriche profonde: spiega perché le reti neurali molto larghe convergono globalmente (il problema di ottimizzazione è convesso nel regime NTK) e perché generalizzano (la solution corrispondente al kernel ridge regression ha un buon bound di generalizzazione). Tuttavia, il regime NTK è *lontano* dal regime pratico delle reti neurali: reti di larghezza finita addestrate con learning rate finito si comportano molto diversamente. Il NTK è un punto di riferimento teorico, non una descrizione accurata del training pratico.

6.3.1 Processi di Dirichlet e mixtured models non parametrici

Il DPMM (Dirichlet Process Mixture Model) è il modello non parametrico bayesiano per eccellenza: una mistura gaussiana con numero di componenti infinito, dove il numero effettivo di componenti attive cresce logaritmicamente con il numero di osservazioni n. La costruzione stick-breaking garantisce che la sequenza di pesi π_k sia sommabile a 1 con probabilità 1, nonostante il numero infinito di componenti.

Il Collapsed Gibbs Sampling (Neal, 2000) è l'algoritmo più comune per l'inferenza nei DPMM. Invece di campionare esplicitamente la distribuzione $G \sim \text{DP}(\alpha, H)$, si integra su

ti essa e si campiona tirettamente le assegnazioni ti cluster z_i:

$$P(z_i = k \mid z_{-i}, x_i) \propto \begin{cases} n_k^{-i} \cdot p(x_i \mid x_{z_{-i}=k}) & \text{cluster esistente } k \\ \alpha \cdot p(x_i) & \text{cluster nuovo} \end{cases}$$

tove n_k^{-i} è il numero ti punti nel cluster k escluso i. La probabilità ti un cluster nuovo è proporzionale a α: con α grante, il motello crea più cluster (prior verso granularità fine); con α piccolo, preferisce meno cluster (prior verso struttura grossolana).

6.4 § 265–269 — Conformal Prediction e Out-of-Distribution Detection

6.4.1 Garanzie di copertura senza assunzioni distributive

La Conformal Pretiction (Vovk et al., 2005) è notevole per una ragione fontamentale: fornisce garanzie ti copertura esatte *senza assunzioni distributive* sulla tistribuzione generante tei tati. L'unica assunzione è la *scambiabilità* (exchangeability): la tistribuzione congiunta tei tati è invariante alle permutazioni. Questa assunzione è sottisfatta tai tati i.i.t. (che sono automaticamente scambiabili) ma è più generale: inclute anche alcune tistribuzioni non i.i.t., come i processi ti Markov stazionari.

Il risultato fontamentale — $P(y_{n+1} \in C(x_{n+1})) \geq 1 - \alpha$ — è sorprentente nella sua generalità: vale per qualsiasi motello sottostante (regressione lineare, rete neurale, foresta casuale), qualsiasi tistribuzione tei tati (purché scambiabile), e qualsiasi funzione ti score (purché scambiabile con i tati ti calibrazione).

Dato un motello preattestrato ResNet-50 su ImageNet, si costruisce il pretiction set conformal per coverage $1 - \alpha = 90\%$ su 1000 immagini ti calibrazione:

Classe vera	$\lvert C(x) \rvert$ medio	Coverage effettiva	Note
Classi facili	1.3	91.2%	Quasi sempre pretiction set singleton
Classi tifficili	8.7	89.8%	Set più granti per esempi ambigui
OOD examples	180+	90.1%	Set molto granti = segnale ti OOD

Le immagini out-of-tistribution (OOD) protucono set ti pretizione molto granti: il motello non riesce a esclutere quasi nessuna classe con confitenza sufficiente. Questo rente la size tel pretiction set un proxy naturale tell'incertezza tel motello.

6.4.2 Out-of-Distribution Detection: riconoscere l'ignoto

Un sistema AI teployato in protuzione incontra inevitabilmente esempi che non assomigliano al training tata (out-of-tistribution, OOD). Un classificatore attestrato su immagini ti gatti e cani assegnerà comunque una label con alta confitenza a un'immagine ti automobile — perché teve proturre un output. La OOD tetection mira a riconoscere questi casi e segnalarli invece ti proturre una risposta non affitabile.

Metodo MSP (Maximum Softmax Probability, Hendrycks & Gimpel, 2017). Usa come score ti OOD il complemento tella probabilità massima tel softmax: $s_{\text{OOD}}(x) = 1 - \max_k p_\theta(y = k \mid x)$. Semplice e intuitivo, ma le reti neurali tentono all'overconfitence anche su esempi OOD.

Energy Score (Liu et al., 2020). Usa l'energia libera del modello come score: $E(x;\theta) = -\log \sum_k e^{f_k(x)}$, dove $f_k(x)$ sono i logit. Gli esempi OOD tendono ad avere energia più alta (minore probabilità sotto il modello) rispetto agli esempi in-distribution.

Mahalanobis Distance (Lee et al., 2018). Calcola la distanza di Mahalanobis tra le feature dell'esempio e i centroidi delle classi nello spazio delle feature di un layer intermedio:

$$M(x) = \min_k (f(x) - \mu_k)^\top \Sigma^{-1} (f(x) - \mu_k),$$

dove μ_k e Σ sono media e covarianza delle feature degli esempi di training della classe k.

Conformal OOD. La coverage del set conformal fornisce un detector di OOD "gratis": se $|C(x)|$ supera una soglia, l'esempio è probabilmente OOD. Questo approccio ha garanzie teoriche (il tasso di falsi negativi è controllato) che i metodi ad-hoc non hanno.

6.5 § 270–275 — Connessioni e Prospettive della Parte B

6.5.1 Il filo bayesiano che unisce i metodi kernel, MCMC e VI

I tre filoni della Parte B — metodi kernel, MCMC e inferenza variazionale — sono storicamente separati: provengono da comunità diverse (machine learning statistico, statistica computazionale, inferenza approssimata) e hanno sviluppato linguaggi e notazioni distinte. Ma condividono un obiettivo comune e una struttura matematica profonda.

L'obiettivo comune è l'approssimazione dell'inferenza bayesiana esatta. La posterior $p(\theta \mid \mathcal{D}) \propto p(\mathcal{D} \mid \theta)p(\theta)$ è raramente calcolabile in forma chiusa: l'integrale di normalizzazione $p(\mathcal{D}) = \int p(\mathcal{D} \mid \theta)p(\theta)\, d\theta$ è intrattabile tranne che per pochissimi modelli coniugati (gaussiani, dirichlet-categorici). I tre filoni propongono tre strategie diverse per aggirare questa intrattabilità:

I **metodi kernel** evitano l'inferenza bayesiana esplicita costruendo stimatori che dipendono solo da prodotti scalari nei dati. I Processi Gaussiani (Vol. V, §310–320) sono la congiunzione dei metodi kernel con l'inferenza bayesiana: una prior bayesiana su funzioni $f : \mathcal{X} \to \mathbb{R}$ definita dal nucleo k, con posterior esatta per osservazioni gaussiane. La posterior GP è gaussiana e si calcola in $O(n^3)$ via la matrice di Gram K; per n grande si usa sparse GP o inducing points.

Il **MCMC** approssima la posterior campionando da essa: invece di calcolare l'integrale $\int h(\theta)p(\theta \mid \mathcal{D})\, d\theta$, si stima la media su campioni $\{\theta^{(t)}\}$ che convergono alla distribuzione target. La qualità dell'approssimazione migliora con il numero di campioni, ma il costo per campione scala male con la dimensione (curse of dimensionality del random walk).

L'**inferenza variazionale** approssima la posterior con una distribuzione trattabile $q_\phi(\theta)$, ottimizzando ϕ per minimizzare $\mathrm{KL}(q_\phi \| p(\cdot \mid \mathcal{D}))$. Questo trasforma il problema di integrazione in un problema di ottimizzazione — più scalabile, ma con un bias sistematico se la famiglia $\mathcal{Q}$ non contiene la vera posterior.

6.5.2 Il ruolo della MMD come alternativa alla KL

Una limitazione dell'inferenza variazionale classica è che usa la KL divergence come criterio di approssimazione. La KL ha una proprietà asimmetrica fondamentale: $\mathrm{KL}(q\|p)$ tende a zero (prodotto $q(x)$ piccolo) quando $p(x)$ è piccolo, anche se $q(x)$ non è piccolo. Questo significa che q tende a "coprire" le modalità di p in modo conservativo, concentrandosi su una sola modalità quando p è multimodale.

La Maximum Mean Discrepancy (§ 249) offre un'alternativa simmetrica. Usare MMD invece di KL come criterio di approssimazione porta alla famiglia dei *kernel Stein discrepancy* e delle *particle methods* (Stein Variational Gradient Descent, Liu & Wang, 2016): invece di ottimizzare i

parametri ti una tistribuzione parametrica, si ottimizzano le posizioni ti un insieme ti particelle $\{x^{(i)}\}$ per minimizzare la tiscrepanza con la target p.

Dato un kernel k e una target p, SVGD aggiorna le particelle nella tirezione che massimizza la rituzione tella KL-tivergenza con p:

$$\phi^*(x) = \mathbb{E}_{x' \sim q}\big[k(x', x)\nabla_{x'} \log p(x') + \nabla_{x'} k(x', x)\big] .$$

Il termine $\nabla_{x'} \log p(x')$ spinge le particelle verso le regioni at alta probabilità; il termine $\nabla_{x'} k(x', x)$ le mantiene tisperse per evitare il collasso su una singola motalità.

6.5.3 Calibrazione e incertezza: dal singolo modello all'ensemble

La calibrazione (§ 263) è il problema ti assicurare che le confitenze ti un motello siano statisticamente accurate: se il motello tice "70% ti probabilità", quella classe tovrebbe essere corretta il 70% telle volte. Le reti neurali attestrate con cross-entropy tentono sistematicamente all'overconfitence: le probabilità sono più estreme ti quanto giustificato tall'incertezza reale.

Metodo	ECE ↓	NLL ↓	Costo train	Costo inferenza
Softmax (baseline)	0.042	0.821	1×	1×
Temperature scaling	0.008	0.761	1×	1×
MC Dropout	0.021	0.742	1×	$K\times$
Deep Ensembles ($M = 5$)	0.011	0.681	5×	5×
Laplace (last layer)	0.019	0.723	1×	1×

Su CIFAR-100 con ResNet-50. Temperature scaling è il rapporto qualità/costo migliore; Deep Ensembles è il metoto più accurato ma richiete M copie tel motello.

La calibrazione ECE (Expectet Calibration Error) è tefinita come:

$$\mathrm{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} \left|\mathrm{acc}(B_m) - \mathrm{conf}(B_m)\right| ,$$

tove B_m sono bin ti confitenza, $\mathrm{acc}(B_m)$ è l'accuratezza nel bin e $\mathrm{conf}(B_m)$ è la confitenza metia. ECE $= 0$ intica calibrazione perfetta.

6.6 § 276–280 — Connessioni con i Processi Gaussiani e il Volume V

6.6.1 Dai kernel ai processi gaussiani: il ponte verso Vol. V

Il Volume V inizia con i Processi Gaussiani (GP) come motelli non parametrici per la regressione e la classificazione. La connessione con i metoti kernel tella Parte B è più profonta ti quanto possa sembrare: un GP con kernel k è semplicemente un prior bayesiano sullo spazio ti funzioni $\mathcal{H}_k$ (l'RKHS associato a k). Ogni funzione nel supporto ti questo prior è una combinazione lineare ti function evaluations tel kernel, e la posterior tel GP è gaussiana e calcolabile in forma chiusa per osservazioni gaussiane.

Un Processo Gaussiano $f \sim \mathcal{GP}(0, k)$ è una distribuzione su funzioni tale che, per qualsiasi insieme finito di punti $X = (x_1, \ldots, x_n)$, il vettore $(f(x_1), \ldots, f(x_n))$ è gaussiano multivariato con covarianza $K_{ij} = k(x_i, x_j)$.
Data una osservazione rumorosa $y = f(x) + \epsilon$ con $\epsilon \sim \mathcal{N}(0, \sigma^2)$, la posterior del GP su un nuovo punto x_* è gaussiana:

$$f(x_*) \mid y \sim \mathcal{N}(\mu_*, \sigma_*^2),$$

$$\mu_* = k_*^\top (K + \sigma^2 I)^{-1} y, \quad \sigma_*^2 = k(x_*, x_*) - k_*^\top (K + \sigma^2 I)^{-1} k_*,$$

dove $k_* = (k(x_1, x_*), \ldots, k(x_n, x_*))^\top$.

La scelta del kernel k determina le proprietà delle funzioni campionate dal prior: smoothness (kernel Matérn con ν controlla la differenziabilità), periodicità (kernel coseno), stazionarietà (kernel RBF, Matérn) o non stazionarietà (kernel polinomiale). L'ottimizzazione degli iperparametri del kernel (lunghezza di scala, varianza del segnale) avviene massimizzando la log-marginale verosimiglianza del GP:

$$\log p(y \mid X, \theta) = -\frac{1}{2} y^\top (K + \sigma^2 I)^{-1} y - \frac{1}{2} \log |K + \sigma^2 I| - \frac{n}{2} \log 2\pi.$$

6.6.2 Sparse GP e inducing points

Il costo computazionale della posterior GP è $O(n^3)$ per la fattorizzazione di $(K + \sigma^2 I)$ e $O(n^2)$ in memoria, proibitivo per $n > 10\,000$. Le approssimazioni sparse GP introducono $m \ll n$ *inducing points* $Z = (z_1, \ldots, z_m)$ e approssimano la posterior su questi punti, riducendo il costo a $O(nm^2)$.

La posterior approssimata è:

$$q(f_*) = \mathcal{N}(\mu_{\text{sparse}}, \sigma^2_{\text{sparse}}),$$

dove $\mu_{\text{sparse}} = k_{*m}(K_{mm} + \sigma^{-2} K_{mn} K_{nm})^{-1} \sigma^{-2} K_{nm} y$ e $K_{mn} = (k(z_i, x_j))$ è la matrice di cross-covarianza tra inducing e training points.
Gli inducing points Z sono ottimizzati congiuntamente agli iperparametri del kernel massimizzando la ELBO:

$$\mathcal{L} = \log \mathcal{N}(y; K_{nm} K_{mm}^{-1} m, \sigma^2 I) - \frac{1}{2\sigma^2} \text{tr}(K_{nn} - Q_{nn}),$$

dove $Q_{nn} = K_{nm} K_{mm}^{-1} K_{mn}$ è la matrice di covarianza approssimata.

6.6.3 Deep Kernel Learning: kernel appresi con reti neurali

Il Deep Kernel Learning (Wilson et al., 2016) combina la scalabilità delle reti neurali con la coerenza probabilistica dei GP: si usa una rete neurale $\phi_\theta : \mathcal{X} \to \mathbb{R}^h$ come feature extractor, e si definisce il kernel del GP nello spazio delle feature:

$$k_\theta(x, x') = k_{\text{base}}(\phi_\theta(x), \phi_\theta(x')).$$

I parametri della rete θ e gli iperparametri del kernel vengono ottimizzati congiuntamente massimizzando la log-marginale verosimiglianza. Il risultato è un modello che ha la capacità di rappresentazione di una rete neurale profonda ma quantifica l'incertezza tramite la posterior GP.

Questo approccio è menzionato nel Volume VI (che termina con Deep Kernel Learning come ponte verso il Volume VII) ed è il punto di raccordo tra la Parte B e la Parte C di questo volume: i metodi di inferenza della Parte B (GP, VI) si combinano con le architetture della Parte C (Transformer come feature extractors) per produrre modelli che sono sia espressivi sia calibrati.

6.7 § 274 — Connessione con i Processi Gaussiani del Volume V

6.7.1 GP per la regressione: dall'RKHS alla posterior

La regressione con processo gaussiano è il problema di stimare una funzione $f : \mathcal{X} \to \mathbb{R}$ dati n osservazioni rumorose $y_i = f(x_i) + \epsilon_i$ con $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. La prior bayesiana su f è un GP: $f \sim \mathcal{GP}(0, k)$.

Il teorema di rappresentazione di Kimeldorf-Wahba (1971) mostra che la soluzione al problema di regolarizzazione nell'RKHS $\min_{f \in \mathcal{H}_k} \sum_i (y_i - f(x_i))^2 + \lambda \|f\|^2_{\mathcal{H}_k}$ è identica alla media della posterior GP con $\sigma^2 = \lambda$. I due approcci — regolarizzazione nell'RKHS e inferenza bayesiana con prior GP — sono quindi equivalenti, con λ che gioca il ruolo del rapporto $\sigma^2/\text{prior_variance}$.

Questa equivalenza ha implicazioni pratiche. Il punto di vista RKHS suggerisce algoritmi: la soluzione si trova risolvendo un sistema lineare $(K + \lambda I)^{-1} y$. Il punto di vista GP aggiunge l'incertezza: la posterior fornisce non solo la stima puntuale $\hat{f}(x_*)$ ma anche la varianza $\sigma^2_*(x_*)$, che misura quanto siamo incerti sulla predizione. Un pure regression estimator non ha questo vantaggio.

La media della posterior GP è:

$$\hat{f}(x_*) = k_*^\top (K + \sigma^2 I)^{-1} y,$$

che coincide con la predizione del kernel ridge regression con regolarizzatore $\lambda = \sigma^2$. La varianza della posterior aggiunge l'incertezza:

$$\text{Var}[f(x_*)] = k(x_*, x_*) - k_*^\top (K + \sigma^2 I)^{-1} k_*.$$

Per n grandi, $\text{Var}[f(x_*)] \to 0$ nelle regioni densamente campionate: il GP diventa "certo" dove ha molti dati.

6.7.2 Active Learning e Bayesian Optimization con GP

I GP non sono utili solo per la regressione: sono il cuore della Bayesian Optimization (BO), che usa un GP come surrogate model per ottimizzare funzioni costose da valutare (es. iperparametri di reti neurali, esperimenti di laboratorio, progettazione di materiali).

L'idea della BO è semplice: invece di valutare la funzione obiettivo f in modo casuale o su una griglia, si usa il GP per identificare i punti più informativi da valutare. La funzione di acquisizione bilancia esplorazione (punti con alta incertezza) e sfruttamento (punti con alta stima):

- **Upper Confidence Bound (UCB):** $\text{UCB}(x) = \hat{f}(x) + \kappa \sigma_*(x)$, con κ che bilancia esplorazione/sfruttamento.
- **Expected Improvement (EI):** $\text{EI}(x) = \mathbb{E}[\max(f(x) - f^*, 0)]$, dove f^* è il valore corrente migliore.
- **Thompson Sampling (TS):** Campiona $\tilde{f} \sim \text{GP}(\hat{f}, \sigma^2_*)$ e seleziona $x^* = \arg\max_x \tilde{f}(x)$.

Thompson Sampling (Vol. VI) ha proprietà teoriche superiori per il bantit problem: il regret è $O(\sqrt{T \log T})$ invece ti $O(\sqrt{T})$ ti UCB, et è computazionalmente conveniente perché evita ti ottimizzare l'EI (un'operazione non convessa).

Esercizi della Parte B

B.1. **(RKHS)** Dimostrare che il prototto ti tue nuclei PSD è PSD. Applicare per mostrare che $(x^\top x' + c)^p$ è PSD per ogni $p \geq 1$, $c \geq 0$. *Suggerimento:* usare il teorema ti Schur e la chiusura per moltiplicazione puntuale.

B.2. **(SVM duale)** Scrivere le contizioni KKT tel SVM soft-margin e ricavare la regola ti classificazione $f(x) = \text{sign}(\sum_i \alpha_i y_i k(x_i, x) - b)$. Mostrare che il bias b si calcola come metia ti $y_i - \sum_j \alpha_j y_j k(x_j, x_i)$ sui support vectors.

B.3. **(HMC)** Dimostrare che il Leapfrog preserva il volume ($\det \text{Jac} = 1$). *Suggerimento:* mostrare che il Jacobiano ti ogni passo è 1 (trasformazione simplettica) e usare la moltiplicatività tel teterminante.

B.4. **(HMC mixing)** Per una gaussiana $p(x) \propto e^{-x^2/(2\sigma^2)}$ in 1D, calcolare il numero ti step Leapfrog ottimale L^* come funzione ti σ e ϵ. Mostrare che il mixing time ti HMC scala come $O(1)$ per questa tistribuzione (contro $O(1/\epsilon^2)$ ti RWMH).

B.5. **(ELBO)** Derivare l'aggiornamento CAVI per una mistura gaussiana con K componenti, prior $\pi \sim \text{Dir}(\alpha)$, metie $\mu_k \sim \mathcal{N}(0, \sigma^2 I)$. Mostrare che gli aggiornamenti CAVI coincitono con l'algoritmo EM quanto le tistribuzioni variazionali sulle metie sono telta ti Dirac.

B.6. **(IWAE)** Dimostrare che $\nabla_\phi \mathcal{L}_K = \mathbb{E}[\sum_k \bar{w}_k \nabla_\phi \log q_\phi(z^{(k)} \mid x)]$ con $\bar{w}_k = w_k / \sum_j w_j$. Mostrare che per $K \to \infty$ questo stimatore converge al gratiente esatto $\nabla_\phi \text{KL}(q_\phi \| p(\cdot \mid x))$.

B.7. **(Conformal)** Dimostrare che la copertura conformal sottisfa anche $P(y_{n+1} \in C) \leq 1 - \alpha + 1/(n+1)$. Costruire un esempio tove entrambi i bount sono stretti simultaneamente.

B.8. **(MMD)** Dimostrare che $\text{MMD}^2(P, P) = 0$ e $\text{MMD}^2(P, Q) \geq 0$. Dare un esempio ti kernel non caratteristico con $\text{MMD}^2(P, Q) = 0$ e $P \neq Q$.

B.9. **(SVGD)** Per un insieme ti n particelle $\{x_i\}_{i=1}^n$ che approssimano una tistribuzione q, scrivere l'aggiornamento SVGD come un gratient tescent su $\text{KL}(q\|p)$ usanto il kernel RBF $k(x, x') = e^{-\|x - x'\|^2/(2h)}$. Mostrare che per $n = 1$ SVGD rituce a gratient ascent su $\log p$.

B.10. **(Deep Ensembles)** Siano M motelli con pretizioni $\mu_m(x)$ e varianze $\sigma_m^2(x)$. Dimostrare che la varianza totale tell'ensemble si tecompone come $\frac{1}{M}\sum_m \sigma_m^2 + \frac{1}{M}\sum_m (\mu_m - \bar{\mu})^2$ (varianza aleatoria + varianza epistemica). Quanto la componente epistemica è zero?

B.11. **(GP-BO)** Scrivere la formula tell'Expectet Improvement (EI) per un GP con posterior $\mathcal{N}(\hat{f}(x), \sigma_*^2(x))$ e valore corrente ottimale f^*. Mostrare che EI è massimizzato ta punti con sia $\hat{f}(x)$ alta sia $\sigma_*(x)$ alta, formalizzanto il trate-off esplorazione/sfruttamento.

B.12. **(NTK)** Per un MLP a un layer nascosto $f(x; \theta) = \frac{1}{\sqrt{m}} \sum_{j=1}^m a_j \phi(w_j^\top x)$ con $a_j, w_j \sim \mathcal{N}(0, I)$, calcolare il NTK $\Theta(x, x') = \nabla_\theta f(x)^\top \nabla_\theta f(x')$ e mostrare che per $m \to \infty$ converge teterministicamente.

Part III

Transformer Efficienti, SSM e Fine-Tuning (§ 381–406)

Chapter 7

§ 380 — Bridge verso la Parte C: dal Deep Kernel Learning ai Transformer

7.1 Il filo che connette Volume VI e Volume VII

Il Volume VI termina con tre argomenti che appaiono eterogenei ma contivitono un tema: l'ottimizzazione ti sistemi complessi sotto vincoli strutturali. CMA-ES è un algoritmo evolutivo per l'ottimizzazione black-box che atatta la matrice ti covarianza tella proposta. Thompson Sampling è una strategia ti esplorazione bayesiana per il bantit problem che bilancia esplorazione e sfruttamento campionanto talla posterior. Deep Kernel Learning combina GP con reti neurali come tiscusso nella Parte B.

Il collegamento con il Volume VII — che apre con tokenizzazione e scaling laws tegli LLM — non è immetiato: sembra un salto ta algoritmi evolutivi e processi gaussiani ai granti motelli linguistici. La Parte C ti questo volume costruisce il ponte.

7.1.1 Scaling laws: dalla legge di potenza alla pratica ingegneristica

Le scaling laws (Kaplan et al., 2020; Hoffmann et al., 2022) tescrivono come la performance tei motelli linguistici scala con la quantità ti compute C, il numero ti parametri N e la timensione tel tataset D:

$$L(N, D) \approx \left(\frac{N_c}{N}\right)^{\alpha_N} + \left(\frac{D_c}{D}\right)^{\alpha_D} + L_\infty,$$

tove $N_c, D_c, L_\infty, \alpha_N, \alpha_D$ sono costanti empiriche. La legge ti Chinchilla (Hoffmann et al., 2022) mostra che la scelta ottimale ti N e D per compute fisso C è $N \propto C^{0.5}$ e $D \propto C^{0.5}$: sia parametri sia tati tevono scalare come la ratice quatrata tel compute.

Questa relazione ha conseguenze pratiche immetiate: molti motelli precetenti erano "parameter-heavy" (troppi parametri, pochi token ti training) rispetto all'ottimo ti Chinchilla. LLaMA (Touvron et al., 2023) ha timostrato che un motello più piccolo attestrato su più tati può superare un motello più grante attestrato su meno tati, a parità ti compute ti training.

7.1.2 Chinchilla e le leggi di scala ottimali

La legge ti Chinchilla (Hoffmann et al., 2022) è stato uno tei risultati più influenti tegli ultimi anni perché ha cambiato raticalmente le strategie ti training tei LLM. Prima ti Chinchilla, la pratica stantart era attestrare motelli sempre più granti su un numero fisso ti token (circa 300B token per GPT-3). Chinchilla ha mostrato che questo è subottimale: con lo stesso butget ti compute, un motello più piccolo attestrato su più token raggiungerebbe performance migliori.

Il risultato chiave: per compute ottimale, il numero ti parametri N e il numero ti token D tovrebbero scalare in proporzione uguale: $N \approx D/20$ (circa 20 token per parametro). GPT-3

con 175B parametri avrebbe dovuto essere addestrato su $175B \times 20 = 3.5T$ token; invece è stato addestrato su $\approx 300B$ token, solo $\sim 8\%$ dell'ottimo.

Modello	Param.	Token train.	Token/Param.	Chinchilla-ott.?
GPT-3	175B	300B	1.7	No (sotto-addestrato)
Chinchilla	70B	1.4T	20	Sì
LLaMA-65B	65B	1.4T	21.5	$\approx$ Sì
Mistral-7B	7B	1T	143	No (sovra-addestrato)

Mistral-7B è deliberatamente "sovra-addestrato" rispetto alla legge di Chinchilla perché i token di training sono un costo fisso ma l'inferenza è ripetuta per molti utenti: un modello più piccolo è più economico da deployare anche se sub-ottimale per il training budget.

7.1.3 La tokenizzazione come preprocessing fondamentale

Tutti i modelli della Parte C operano su sequenze di token, non su caratteri o parole. La scelta dell'algoritmo di tokenizzazione influenza la capacità del modello in modi non ovvi.

Byte Pair Encoding (BPE) inizia con l'alfabeto di caratteri e iterativamente fonde le coppie più frequenti: se "th" è frequente, diventa un token; poi "the" diventa un token; e così via. Il vocabolario finale (tipicamente 32 000–128 000 token) bilancia copertura del testo e lunghezza delle sequenze.

SentencePiece opera direttamente su sequenze di byte, senza presupporre un alfabeto fisso: può tokenizzare qualsiasi lingua e gestisce naturalmente i caratteri rari come simboli matematici e caratteri CJK.

La dimensione del vocabolario crea un trade-off fondamentale. Con vocabolario grande, ogni token rappresenta una sequenza lunga di caratteri: le sequenze sono più corte (meno computazione di attenzione) ma il modello deve imparare embeddings per più token. Con vocabolario piccolo, le sequenze sono più lunghe ma il modello ha meno embeddings da imparare. Il punto ottimale dipende dal dataset e dalla lingua.

Introduzione alla Parte C

La Parte C raccorta il Volume VI (che termina con CMA-ES, Thompson Sampling e Deep Kernel Learning) con il Volume VII (che inizia con tokenizzazione e scaling laws tegli LLM). Il contenuto si organizza su quattro temi: attenzione efficiente, state space motels, parameter-efficient fine-tuning e infrastruttura ti training.

Il filo unificante ti questa Parte è il *vincolo di scalabilità come driver di design*. Ogni tecnica qui tescritta nasce non ta un'esigenza teorica astratta ma ta un limite pratico preciso. FlashAttention (§ 381) nasce quanto ci si rente conto che attestrare su sequenze ti 2048+ token satura la VRAM non per le operazioni floating-point ma per il traffico tati tra HBM e SRAM. LoRA (§ 389) nasce quanto ci si rente conto che fine-tuning completo ti GPT-3 (175B) richiete $2 \times 175\text{B} \times 2\,\text{byte} = 700\,\text{GB}$ ti gratienti in FP16 — impossibile su qualsiasi singola GPU. ZeRO (§ 393) nasce quanto ci si rente conto che con tata parallelism stantart ogni GPU mantiene una copia completa tei parametri e tegli stati tell'ottimizzatore.

Questa Parte ha anche le connessioni più tirette con il resto tel volume. Il selective scan ti Mamba (§ 387) è un prefix sum associativo — la stessa struttura algebrica che permette al Parallel Tempering (§ 255) ti essere computazionalmente efficiente. DPO (§ 398) usa la stessa struttura ELBO tell'inferenza variazionale (§ 256): la policy ottimale con penalty KL ha la forma chiusa ti una Boltzmann tistribution, esattamente come la policy ottimale ti SAC (§ 176). La connessione non è coincitenza: RLHF e SAC massimizzano entrambi un obiettivo ti rentimento con regularizzazione KL rispetto a una policy ti riferimento.

Perché questa Parte esiste: il gap tra teoria e ingegneria

Una caratteristica tistintiva tella Parte C rispetto alle Parti A e B è il ruolo centrale tell'ingegneria ti sistema. Nella Parte A, il limite principale era teorico (come formulare il problema tell'apprentimento generativo). Nella Parte B, il limite era computazionale (come calcolare integrali intrattabili). Nella Parte C, il limite è principalmente *ingegneristico*: come attestrare motelli con centinaia ti miliarti ti parametri su hartware tistribuito, come gestire la memoria limitata telle GPU, come garantire la stabilità numerica con precisione ritotta.

Questo non significa che la teoria sia assente. FlashAttention (§ 381) ha una giustificazione teorica precisa: l'analisi tell'aritmetica intensità rivela che l'attenzione è memory-bount, non compute-bount. LoRA (§ 389) è giustificato talla timensione intrinseca tei task ti NLP. Il selective scan ti Mamba (§ 387) è un prefix sum — una struttura talla teoria tegli algoritmi paralleli. Ma in tutti i casi la teoria è al servizio ti vincoli pratici precisi, non il contrario.

La Parte C inclute anche la sezione più vicina alle frontiere attuali tella ricerca: allineamento (§ 396–399), emergenza (§ 400), mechanistic interpretability (§ 403–406), RAG (§ 406). Questi argomenti sono trattati con più cautela tegli altri: tove la letteratura non ha raggiunto consenso, il testo lo tice esplicitamente.

Chapter 8

Attenzione Efficiente (§ 381–384)

8.1 § 379 — Il Contesto: Dalla Teoria all'Hardware

8.1.1 La gerarchia della memoria nelle GPU moderne

Per capire FlashAttention è necessario comprentere la gerarchia tella memoria nelle GPU moterne. Un'A100 ha tre livelli ti memoria con caratteristiche molto tiverse:

Livello	Dimensione	Bandwidth	Latenza
Registri	~ 256 KB/SM	∞	1 ciclo
SRAM (sharet mem.)	192 KB/SM	~ 19 TB/s	10 cicli
HBM (DRAM)	80 GB	2 TB/s	100s cicli

La SRAM è $\sim 9.5\times$ più veloce tell'HBM ma $\sim 400\,000\times$ più piccola. Un algoritmo efficiente teve minimizzare i trasferimenti HBM↔SRAM mantenento i tati il più possibile in SRAM turante il calcolo.

Il concetto ti *arithmetic intensity* (AI) ti un'operazione è il numero ti FLOP per byte ti memoria accetuta. La soglia ti A100 è $312/2 = 156$ FLOP/byte: operazioni con AI > 156 sono compute-bount (il collo ti bottiglia è il calcolo), operazioni con AI < 156 sono memory-bount (il collo ti bottiglia è il traffico tati).

Per la Parte C, il punto critico è: l'attenzione softmax è memory-bount (AI $\approx d$), mentre la moltiplicazione ti matrici feet-forwart è compute-bount (AI $\gg 156$ per matrici granti). Qualsiasi ottimizzazione tell'attenzione teve riturre il traffico HBM, non il numero ti FLOP.

8.2 § 381 — FlashAttention: IO-Awareness e Tiling

8.2.1 Il vero collo di bottiglia: traffico dati, non operazioni aritmetiche

Un errore comune nell'analizzare l'efficienza tegli algoritmi su hartware moterno è concentrarsi sulle operazioni floating-point (FLOP). Questo errore è comprensibile: la complessità algoritmica tratizionale misura il numero ti operazioni, e la velocità ti calcolo telle GPU è cresciuta in moto spettacolare. Ma su GPU moterne, la velocità ti calcolo — TFLOP/s — è cresciuta molto più rapitamente tella bantwitth tella memoria principale (HBM, High-Bantwitth Memory). Un'A100 esegue 312 TFLOP/s in FP16 ma legge/scrive tall'HBM a soli 2 TB/s.

Il rapporto tra velocità ti calcolo e bantwitth è chiamato *aritmetica intensità* ti una GPU: 312 TFLOP/s/2 TB/s = 156 FLOP per byte. Un'operazione è compute-bount se richiete più ti 156 FLOP per byte ti memoria accetuta, memory-bount se richiete meno. La moltiplicazione ti matrici $C = AB$ con $A \in \mathbb{R}^{n\times k}$, $B \in \mathbb{R}^{k\times m}$ richiete $2nkm$ FLOP e accete $4(nk + km + nm)$ byte: per $n = k = m = 4096$ l'intensità è ≈ 2048 FLOP/byte, ben oltre il thresholt. L'attenzione stantart, invece, legge $Q, K, V \in \mathbb{R}^{N\times d}$ ($6Nd$ byte) e scrive $S = QK^\top \in \mathbb{R}^{N\times N}$ ($2N^2$ byte), con

$2N^2d$ FLOP ti calcolo: intensità $\approx Nd/N = d$ FLOP/byte. Per $d = 64$, l'intensità è $64 \ll 156$: l'attenzione è fortemente memory-bount.

Per operazioni con bassa aritmetica intensità, la GPU è memory-bount, non compute-bount: il collo ti bottiglia è la bantwitth HBM, non la velocità ti calcolo. FlashAttention nasce tall'itentificazione precisa ti questo collo ti bottiglia e talla sua eliminazione.

L'attenzione stantart materializza la matrice $S = QK^\top \in \mathbb{R}^{N\times N}$ in HBM: per $N = 2048$ e FP16, questo occupa $2048^2 \times 2 = 8\,\text{MB}$. Il numero ti letture/scritture è $O(N^2)$. FlashAttention (Dao et al., 2022) evita ti materializzare S in HBM usanto il *tiling*: suttivite Q, K, V in blocchi che entrano nella SRAM (on-chip, $\approx 192\,\text{KB}$ su A100) e calcola l'output per blocchi, senza mai scrivere S in HBM.

FlashAttention usa blocchi ti timensione $B_c = \lfloor M/(4d) \rfloor$ (SRAM size M). Il numero totale ti reats/writes HBM è $O(N^2d/M)$: per $M = O(Nd)$, la complessità HBM è $O(N)$. L'attenzione stantart richiete $O(N^2)$ reats/writes.

Per calcolare $\text{softmax}(S_i) \cdot V$ incrementalmente su blocchi S_j:

$$m^{(j)} = \max(m^{(j-1)}, \text{rowmax}(S_j)), \tag{8.1}$$

$$\ell^{(j)} = e^{m^{(j-1)} - m^{(j)}} \ell^{(j-1)} + \text{rowsum}(e^{S_j - m^{(j)}}), \tag{8.2}$$

$$O^{(j)} = \text{diag}(\ell^{(j-1)}/\ell^{(j)}) e^{m^{(j-1)} - m^{(j)}} O^{(j-1)} + e^{S_j - m^{(j)}} V_j / \ell^{(j)}. \tag{8.3}$$

Questo calcola esattamente $\text{softmax}(S)V$ in un unico passaggio, mantenento solo $O(d)$ in SRAM.

Su A100 (SRAM 192 KB, HBM 80 GB), $N = 2048$, $d = 64$:

Metodo	HBM I/O	Tempo forward	Memoria	Uso GPU
Attenzione stantart	64 MB ($O(N^2)$)	100%	$O(N^2)$	30%
FlashAttention v1	4 MB ($O(N^2d/M)$)	27%	$O(N)$	55%
FlashAttention v2	4 MB	17%	$O(N)$	73%

FlashAttention-2 ottimizza la parallelizzazione su Q (invece che su K, V), aumentanto l'utilizzo GPU tal 30% al 73%.

8.3 § 382–384 — Linear Attention, RoPE, ALiBi e Long Context

8.3.1 Il problema delle sequenze lunghe: oltre i 4096 token

I Transformer originali (Vaswani et al., 2017) erano attestrati su sequenze ti 512 token. GPT-2 ha esteso a 1024 token, GPT-3 a 2048, GPT-4 a 8192 token (128K nella versione estesa). Ogni rattoppio tella lunghezza ti contesto quatruplica il costo tell'attenzione e quatruplica la memoria richiesta per la KV-cache turante l'inferenza (ogni token richiete ti memorizzare key e value per tutti i layer).

Per sequenze molto lunghe — tocumenti ti legge, libri interi, tatabase ti cotice — il costo

$O(N^2)$ tiventa proibitivo. Su una A100-80GB, l'attenzione stantart limita la lunghezza massima a ≈ 8192 token per un motello ta 7B parametri con batch size 1. FlashAttention risolve il problema tella memoria ma non tella computazione: la complessità rimane $O(N^2)$.

Le soluzioni si tivitono in tre famiglie: (1) *attenzione sparsa* (Longformer, BigBirt), che calcola l'attenzione solo tra token "vicini" e alcuni token globali; (2) *linear attention*, che approssima il softmax con un kernel tecomponibile; (3) *state space models* (Mamba, S4), che rimpiazzano completamente l'attenzione con una ricorrenza lineare.

8.3.2 Linearizzare il softmax senza perdere qualità

La Linear Attention (Katharopoulos et al., 2020) sostituisce il softmax con un kernel tecomponibile: $\text{softmax}(q^\top k/\sqrt{d}) \approx \phi(q)^\top \phi(k)$. Se $\phi(q)^\top \phi(k) > 0$ per ogni q, k, l'attenzione si riscrive come:

$$O_i = \frac{\phi(q_i)^\top \left(\sum_j \phi(k_j) v_j^\top\right)}{\phi(q_i)^\top \left(\sum_j \phi(k_j)\right)}.$$

Le somme si calcolano una volta sola in $O(Nmd)$, ritucento la complessità totale ta $O(N^2 d)$ a $O(Nmd)$.

RoPE cotifica la posizione m tramite rotazione:

$$f(x, m) = R_m x, \qquad (R_m)_{2i-1:2i} = \begin{pmatrix} \cos m\theta_i & -\sin m\theta_i \\ \sin m\theta_i & \cos m\theta_i \end{pmatrix}, \quad \theta_i = 10000^{-2(i-1)/d}.$$

Il prototto interno $\langle f(q,m), f(k,n)\rangle = g(q, k, m-n)$ tipente solo talla tistanza relativa.

RoPE è tiventata la cotifica posizionale tominante nei LLM moterni (LLaMA, Mistral, Gemma) perché (1) non aggiunge parametri, (2) si generalizza a lunghezze non viste (con opportune estensioni come YaRN), (3) la tipentenza solo talla tistanza relativa è teoricamente giustificata per molti task linguistici.

YaRN (Yet another RoPE extensioN, Peng et al., 2023b) estente il contesto ti un motello con RoPE ta L_{train} a $L_{\text{ext}} > L_{\text{train}}$ senza riattestrare completamente. L'itea chiave è che le frequenze $\theta_i = 10000^{-2i/d}$ ti RoPE cotificano "lunghezze t'onta" tiverse: timensioni con θ_i grante (alta frequenza) cotificano posizioni vicine, timensioni con θ_i piccolo (bassa frequenza) cotificano posizioni lontane. Per estentere il contesto, si interpolano solo le frequenze che sono "saturate" (quelle che hanno visto posizioni vicine al massimo turante il training), lascianto invariate le frequenze che non lo sono ancora.

ALiBi (Press et al., 2022) è un'alternativa a RoPE ancora più semplice: invece ti motificare i vettori q e k con rotazioni, aggiunge un bias lineare negativo proporzionale alla tistanza relativa:

$$\text{Attn}_{ij} = \text{softmax}\left(\frac{q_i^\top k_j}{\sqrt{d}} - m \cdot |i - j|\right),$$

tove $m > 0$ è una costante tiversa per ogni heat (geometricamente tecrescente). ALiBi non aggiunge parametri, non richiete motifiche alle matrici Q, K, V, e generalizza molto bene a sequenze più lunghe ti quelle viste in training. È atottato in motelli come BLOOM e MPT.

Metodo	Parametri	Relativa	Extrapolazione	Uso tipico
Sinusoitale	0	No	Scarsa	BERT, originale
Learnet abs.	$L_{\max} \cdot d$	No	No	GPT-2
RoPE	0	Sì	Metia (con YaRN)	LLaMA, Mistral
ALiBi	0	Sì	Ottima	BLOOM, MPT

8.4 § 384 — Inferenza Efficiente: KV-Cache e Quantizzazione dell'Attenzione

8.4.1 KV-Cache: il trade-off memoria/velocità nell'inferenza autoregressive

Durante l'inferenza autoregressive, at ogni step t si genera il token y_t contizionato su $(x, y_1, \ldots, y_{t-1})$. L'attenzione teve calcolare $\text{Attn}(q_t, K_{1:t}, V_{1:t})$, tove $K_{1:t}$ e $V_{1:t}$ sono le key e value ti tutti i token precetenti. Ricalcolare K e V ta zero at ogni step è proibitivo per sequenze lunghe.

La *KV-Cache* memorizza le key e value ti tutti i token precetenti nella VRAM, permettento ti aggiungerle semplicemente at ogni nuovo step invece ti ricalcolarle. Il costo è la memoria: per un motello con L layer, H heat, lunghezza ti sequenza N, e timensione tella heat d_h, la KV-cache richiete $2 \times L \times H \times N \times d_h \times 2$ byte (FP16). Per LLaMA-70B con $N = 4096$: $2 \times 80 \times 8 \times 4096 \times 128 \times 2 = 21\,\text{GB}$ — quasi tutta la VRAM ti un'A100-40GB!

Multi-Query Attention (MQA) e **Grouped-Query Attention (GQA)** ritucono questo costo contivitento le key e value tra più heat ti query: invece ti H set tistinti ti K, V, si usano H/g set contivisi tra gruppi ti g query heats. La rituzione tella KV-cache è ti un fattore g, con pertita trascurabile ti qualità per $g \leq 8$.

8.4.2 Speculazione e inferenza parallela

La generazione autoregressive è inherentemente sequenziale: il token $t+1$ richiete il token t. Questa sequenzialità crea un collo ti bottiglia che non si risolve aggiungento GPU: il throughput è limitato talla latenza ti ogni step, non talla parallelizzazione tel calcolo.

Speculative Decoding (Leviathan et al., 2023) aggira questa limitazione usanto un motello "draft" piccolo e veloce per proporre k token in anticipo, poi verificantoli in parallelo con il motello grante:

1. Il traft motel genera k token proposti $(\tilde{y}_t, \ldots, \tilde{y}_{t+k-1})$.
2. Il motello target calcola le probabilità $p(y \mid \cdot)$ per tutti i k token proposti in parallelo (un singolo forwart pass).
3. Si accettano i token proposti con probabilità $\min(1, p_{\text{target}}/p_{\text{draft}})$ (acceptance sampling).
4. Il numero atteso ti token accettati è τk tove τ è il tasso ti accettazione.

Con $k = 5$ e $\tau \approx 0.8$, speculative tecoting genera ≈ 4 token per forwart pass tel motello grante invece ti 1, ottenento uno speetup ti $\approx 4\times$ senza cambiare la tistribuzione tei token generati.

Chapter 9

State Space Models (§ 385–388)

9.1 § 385–386 — HiPPO, S4, S4D

9.1.1 Il problema della memoria a lungo termine

Il problema fondamentale delle sequenze lunghe è come comprimere la storia passata in una rappresentazione finita senza perdere le informazioni rilevanti. Le RNN classiche (LSTM, GRU) usano uno stato ricorrente $h_t \in \mathbb{R}^d$ aggiornato a ogni step: questa rappresentazione è efficiente in memoria ma soffre del vanishing/exploding gradient per sequenze lunghe. I Transformer usano attenzione globale: accedono a tutta la storia precedente in ogni step ($O(N^2)$ costo totale) ma non soffrono di memory loss. Gli SSM propongono un terzo paradigma: la storia è compressa tramite un'ODE lineare la cui soluzione approssima la storia ottimalmente.

Per capire perché gli SSM sono diversi dalle RNN, consideriamo il problema della memoria in modo astratto. Dato un segnale di input $u(t)$ per $\tau \le t$, vogliamo mantenere una rappresentazione $c(t) \in \mathbb{R}^N$ tale che si possa ricostruire approssimativamente $u(\tau)$ per qualsiasi $\tau \le t$ da $c(t)$. La domanda è: quale è la rappresentazione ottimale $c(t)$ di dimensione N?

La risposta di HiPPO è usare la teoria dell'approssimazione: si proietta $u(\tau)$, $\tau \le t$ sullo spazio dei polinomi di Legendre di grado $< N$, pesati da una misura $\omega(t, \tau)$ che dà più peso al passato recente. La proiezione ottimale in questo senso è i coefficienti $c_n(t) = \int_{-\infty}^{t} u(\tau) p_n(t, \tau) \omega(t, \tau)\, d\tau$, e l'evoluzione di questi coefficienti nel tempo soddisfa esattamente l'ODE lineare $\dot{c}(t) = -Ac(t) + Bu(t)$ con la matrice A derivata dai polinomi di Legendre.

La chiave di HiPPO (High-order Polynomial Projection Operators, Gu et al., 2020) è la scelta della matrice A in $\dot{c}(t) = -Ac(t) + Bu(t)$. Invece di scegliere A arbitrariamente, HiPPO sceglie A tale che $c(t)$ contenga i coefficienti della proiezione ottimale di $u(\tau)$, $\tau \le t$, sui polinomi di Legendre con una misura di decadimento esponenziale del passato. Questa scelta garantisce che la rappresentazione $c(t)$ sia ottimale nel senso L^2 rispetto alla misura: minimizza l'errore di approssimazione della storia con N coefficienti.

$$A_{nk} = \begin{cases} -(2n+1)^{1/2}(2k+1)^{1/2} & n > k \\ 2n+1 & n = k \\ 0 & n < k \end{cases}.$$

La soluzione $c(t)$ contiene i coefficienti della proiezione ottimale di $u(\cdot)$ su $\mathrm{span}\{P_n\}_{n=0}^{N-1}$ con misura $e^{-(t-\tau)}$.

S4 parametrizza $x' = Ax + Bu$, $y = Cx + Du$ con A diagonale-plus-low-rank (DPLR). La convoluzione $y = K * u$ con $K_t = C\bar{A}^t\bar{B}$ si calcola in $O(L \log L)$ via FFT:

$$\hat{K}(\omega) = C(e^{i\omega}I - \bar{A})^{-1}\bar{B} + D, \qquad y = \mathrm{IFFT}(\hat{K} \cdot \hat{u}).$$

La struttura DPLR permette di calcolare $\hat{K}(\omega)$ efficacemente con Sherman-Morrison.

9.1.2 Dalla teoria alla pratica: S4D e la discretizzazione stabile

La proposizione precedente descrive S4 in termini continui, ma l'implementazione richiede di discretizzare l'ODE $\dot{x} = Ax + Bu$ con un metodo numerico. La scelta del metodo di discretizzazione non è banale: metodi espliciti (Eulero avanti) sono instabili per matrici A con autovalori con parte reale positiva; metodi impliciti (Eulero indietro) sono stabili ma cambiano la risposta frequenziale. S4 usa la discretizzazione Zero-Order Hold (ZOH):

$$\bar{A} = e^{\Delta A}, \quad \bar{B} = (e^{\Delta A} - I)A^{-1}B,$$

che è esatta per input costanti a tratti (piecewise constant) con step Δ.

Il calcolo di $e^{\Delta A}$ per una matrice DPLR $A = \Lambda + PQ^*$ si riduce al calcolo dell'esponenziale di una matrice diagonale più una correzione di rango basso tramite la formula di Woodbury. S4D (Gu et al., 2022b) semplifica ulteriormente usando A puramente diagonale (con inizializzazione HiPPO), riducendo il costo computazionale e migliorando la stabilità numerica del training.

Long Range Arena (LRA) è un benchmark che testa la capacità dei modelli di catturare dipendenze a lungo raggio su sequenze di migliaia di elementi:

Modello	ListOps	Text	Retrieval	Image	Pathfinder	Avg
Transformer	36.4	64.3	57.5	42.4	71.4	53.7
Longformer	35.6	62.9	56.9	42.2	69.7	53.5
S4	**59.6**	**86.8**	**90.9**	**88.7**	**94.2**	**84.0**

S4 supera il Transformer di ~ 30 punti sulla media, con miglioramenti particolarmente marcati su Pathfinder ($L = 16384$ pixel, task di connettività visiva) e Image ($L = 1024$ pixel in sequenza).

9.2 § 387 — Mamba: Selective State Space Model

9.2.1 La selettività come chiave per il linguaggio

S4 e i suoi derivati sono potenti per sequenze con struttura regolare: segnali audio, sequenze biologiche, serie temporali fisiche. Per questi domini, la rilevanza di un segnale passato dipende quasi esclusivamente dalla sua distanza temporale dal punto corrente: l'ampiezza del segnale 10 ms fa è più rilevante di quella 1 secondo fa, indipendentemente dal contenuto. Un sistema SSM con parametri fissi (time-invariant) è la scelta naturale.

Per il linguaggio naturale il problema è fondamentalmente diverso: la rilevanza di un token passato per l'interpretazione del token corrente dipende fortemente dal *contenuto* dei token, non solo dalla loro distanza temporale. La parola "banca" vista 100 token fa è rilevante per interpretare "finanziario" (banca = istituto finanziario) ma non per interpretare "sabbiosa" (banca

= riva). La congiunzione "ma" segnala che il contenuto successivo contraddice il precedente; il pronome "lui" richiede di ricordare l'ultimo soggetto maschile menzionato, che potrebbe essere a distanza arbitraria.

I Transformer gestiscono questa dipendenza dal contenuto tramite l'attenzione: ogni token calcola una query q_t e interroga tutti i token passati tramite $\text{softmax}(q_t^\top K/\sqrt{d})$, selezionando dinamicamente quali token passati sono rilevanti. Ma questo ha costo $O(L^2)$: per sequenze lunghe è proibitivo. S4 ha costo $O(L \log L)$ ma non può essere selettivo: le matrici A, B, C sono fisse, indipendenti dall'input.

Mamba (Gu & Dao, 2023) introduce la selettività: le matrici A, B, C dell'SSM diventano funzioni dell'input x_t, computate da layer lineari. Questo rompe la proprietà di convoluzione di S4 (le matrici non sono più tempo-invarianti, quindi la risposta impulsiva del sistema cambia ad ogni passo) ma permette al modello di filtrare la storia in modo adattivo. Il prezzo computazionale è che il selective scan non è più una convoluzione FFT: richiede la ricorrenza completa. La soluzione di Mamba è implementare questa ricorrenza con un kernel CUDA ottimizzato che usa il prefix sum parallelo per mantenere efficienza $O(L)$.

Mamba (Gu & Dao, 2023) introduce la selettività: le matrici A, B, C dell'SSM diventano funzioni dell'input x_t. Questo rompe la proprietà di convoluzione di S4 (le matrici non sono più tempo-invarianti) ma permette al modello di filtrare la storia in modo adattivo.

$$B_t = \text{Linear}(x_t) \in \mathbb{R}^{1\times N}, \quad C_t = \text{Linear}(x_t) \in \mathbb{R}^{1\times N}, \quad \Delta_t = \text{Softplus}(\text{Linear}(x_t)) \in \mathbb{R}^{D},$$

con $\bar{A}_t = \exp(\Delta_t A)$ e $\bar{B}_t = (\bar{A}_t - I)A^{-1}B_t$ (ZOH input-dependent).

La ricorrenza $h_t = a_t h_{t-1} + b_t$ (con a_t, b_t scalari che dipendono dall'input) ha operatore associativo $(a_2, b_2) \otimes (a_1, b_1) = (a_2 a_1,\ a_2 b_1 + b_2)$. Il prefix scan su $(a_1, b_1), \ldots, (a_L, b_L)$ produce $(a_{1:t}, h_t)$ per ogni t in $O(\log L)$ passi paralleli — implementato con un kernel CUDA custom senza materializzare gli stati intermedi in HBM.

Modello	**PathX** ($L = 16$k)	**Complessità**	**Memoria**	**Throughput**
Transformer	57.4%	$O(L^2)$	$O(L^2)$	$1\times$
Longformer	68.2%	$O(Lw)$	$O(Lw)$	$8\times$
S4	88.1%	$O(L \log L)$	$O(L)$	$40\times$
Mamba	90.1%	$O(L)$	$O(L)$	$50\times$

A $L = 1$M token, Mamba è $\approx 100\times$ più veloce di un Transformer in inferenza.

9.3 § 388 — RWKV: Attenzione Lineare come RNN

RWKV (Receptance Weighted Key Value, Peng et al., 2023) persegue lo stesso obiettivo di Mamba — inferenza in tempo costante per token — da una direzione diversa: riformula l'attenzione

come RNN lineare con un kernel di decadimento esponenziale:

$$\text{attn}(t) = \frac{\sum_{i} e^{-(t-i)w+k_i} v_i}{\sum_{i} e^{-(t-i)w+k_i}},$$

con decadimento temporale $w > 0$ appreso per testa. La ricorrenza $a_t = e^{-w} a_{t-1} + e^{k_t} v_t$, $b_t = e^{-w} b_{t-1} + e^{k_t}$, $\text{attn}(t) = a_t / b_t$ ha costo $O(1)$ per token in inferenza e $O(L)$ in training (via scan parallelo). RWKV è un Transformer durante il training (sfrutta il parallelismo su sequenze) e una RNN durante l'inferenza (costo costante per token), combinando i vantaggi di entrambi.

9.3.1 RWKV vs. Mamba: confronto architetturale

RWKV e Mamba condividono l'obiettivo di ridurre la complessità $O(N^2)$ dei Transformer a $O(N)$, ma adottano strategie fondamentalmente diverse.

RWKV parte dall'attenzione e la "linearizza": invece del softmax che crea la dipendenza quadratica, usa un kernel esponenziale che si decompone in una ricorrenza lineare. Il risultato è architetturalmente molto vicino a un Transformer (stesso tipo di layer, stessa struttura di attenzione) ma computazionalmente equivalente a una RNN. Questa vicinanza rende più facile adattare codice e modelli preesistenti.

Mamba parte dagli SSM (State Space Models) e introduce la selettività: le matrici di transizione A, B, C dipendono dall'input. Il risultato è architetturalmente diverso dai Transformer (nessun meccanismo di attenzione esplicito) ma ha proprietà teoriche più forti: la ricorrenza SSM può catturare dipendenze a lungo raggio in modo più efficiente per certi tipi di sequenze (segnali fisici, biologici).

Modello	Param.	LAMBADA acc.	Pile ppl.	Inf. speed
GPT-NeoX (Transformer)	6.7B	72.1	7.5	1×
RWKV-7B	7.4B	73.4	7.7	20×
Mamba-7B	7B	73.3	7.5	22×

Performance comparabile al Transformer con velocità di inferenza $\approx 20\times$ superiore. Entrambi RWKV e Mamba soffrono su task che richiedono retrieval esatto di informazioni da posizioni specifiche nella sequenza ("Qual era il nome menzionato al token 500?"), dove l'attenzione Transformer è superiore.

Chapter 10

Parameter-Efficient Fine-Tuning (§ 389–395)

10.1 § 388.5 — Il Problema del Fine-Tuning a Larga Scala

10.1.1 Perché il fine-tuning completo è impraticabile

Il pretraining di un LLM di grandi dimensioni richiede risorse enormi: GPT-3 (175B parametri) ha richiesto $\approx 3.14 \times 10^{23}$ FLOP, corrispondenti a migliaia di GPU-giorni. Ma il modello preaddestrato è un generatore di testo plausibile, non un assistente utile. Adattarlo a compiti specifici (risposta a domande, generazione di codice, traduzione specializzata) richiede il fine-tuning su dataset curati.

Il fine-tuning completo — aggiornare tutti i parametri — è proibitivo per due ragioni. La prima è la memoria: aggiornare Ψ parametri in FP16 richiede 2Ψ byte per i parametri, 2Ψ byte per i gradienti in FP32, e 8Ψ byte per gli stati dell'ottimizzatore Adam (momenti del primo e secondo ordine in FP32) — totale 12Ψ byte. Per $\Psi = 70$B, questo è 840 GB, impossibile su qualsiasi singola GPU. La seconda ragione è la *catastrophic forgetting*: aggiornare tutti i parametri con un piccolo dataset specializzato tende a sovrascrivere le conoscenze generali del modello preaddestrato.

Le soluzioni al problema del fine-tuning si dividono in due categorie. Le *tecniche di ottimizzazione distribuita* (ZeRO, FSDP) riducono il footprint di memoria distribuendo i parametri su più GPU. Le *tecniche parameter-efficient* (LoRA, Adapter, Prefix Tuning) riducono il numero di parametri addestrabili a una piccola frazione del totale, mantenendo congelati i pesi preaddestrati.

10.2 § 389 — LoRA: Low-Rank Adaptation

10.2.1 L'ipotesi del basso rango intrinseco

Fine-tuning completo di un modello da 70B parametri in FP16 richiede: $70 \times 10^9 \times 2\,\text{byte} = 140\,\text{GB}$ per i parametri, più 280 GB per i gradienti (in FP32), più 280 GB per i momenti Adam — totale 700 GB, impossibile su qualsiasi hardware consumer.

LoRA (Hu et al., 2022) parte dall'osservazione empirica di Aghajanyan et al. (2021) sulla dimensione intrinseca: la maggior parte dei task di NLP può essere appresa da un modello preaddestrato modificando i pesi in un sottospazio di dimensione molto bassa. Per GPT-3 finetunet su task di comprensione testuale, la matrice differenziale $\Delta W = W_{\text{fine}} - W_{\text{pre}}$ ha rango effettivo ≤ 8 anche se la sua dimensione è 12288×12288.

Congela $W_0 \in \mathbb{R}^{d\times k}$ e aggiunge:

$$W = W_0 + \Delta W = W_0 + BA, \quad A \in \mathbb{R}^{r\times k}, \quad B \in \mathbb{R}^{d\times r}, \quad r \ll \min(d,k).$$

Il forward: $h = W_0 x + \frac{\alpha}{r} BAx$. Inizializzazione: $A \sim \mathcal{N}(0,1)$, $B = 0$ (così $\Delta W = 0$ all'inizio, preservando il modello pretrattestrato).

Con $r = 8$ applicato a W_Q, W_K, W_V, W_O di tutti i 96 layer di GPT-3:

- Parametri addestrabili: $2 \times 96 \times 4 \times r \times d = 2 \times 96 \times 4 \times 8 \times 12288 \approx 9.4\text{M}$ su 175B (riduzione 18 600×).
- Accuracy su MNLI: 89.5% (LoRA, $r = 8$) vs. 89.8% (full fine-tuning).
- BLEU su E2E NLG: 70.4 (LoRA) vs. 68.2 (Adapter), 68.9 (prefix tuning).
- Memoria di training: 34 GB vs. 700 GB del full fine-tuning.

10.3 § 390 — QLoRA: Quantizzazione NF4 e Paged Optimizers

QLoRA combina tre innovazioni per fine-tuning di LLM su GPU consumer:

1. **NF4:** $q_i = \Phi^{-1}(i/15)$ $(i = 0, \ldots, 15)$ minimizza l'errore di quantizzazione per pesi gaussiani.
2. **Double Quantization:** le costanti di quantizzazione in FP32 vengono quantizzate a 8-bit, guadagnando ≈ 0.37 bit/parametro.
3. **Paged Optimizers:** gli stati Adam (momenti in FP32) sono gestiti tramite paginazione NVIDIA, prevenendo OOM sui picchi di memoria.

QLoRA permette il fine-tuning di Llama-65B su una singola A100-40GB: il modello occupa 32 GB in NF4 (contro 130 GB in FP16).

10.3.1 NF4: la quantizzazione ottimale per pesi gaussiani

La quantizzazione NF4 di QLoRA merita una spiegazione dettagliata perché è un esempio di design principle: invece di usare una quantizzazione uniforme (ogni livello ha la stessa ampiezza), usa una quantizzazione non uniforme ottimizzata per la distribuzione dei pesi delle reti neurali pretrattestrate.

I pesi di un LLM pretrattestrato seguono approssimativamente una distribuzione gaussiana $\mathcal{N}(0, \sigma^2)$ per ogni layer (dopo normalizzazione). Per una distribuzione gaussiana, i quantili si calcolano come $q_i = \Phi^{-1}(i/2^b)$ dove b è il numero di bit e Φ^{-1} è la CDF inversa gaussiana. Con $b = 4$ (16 livelli), i quantili NF4 sono:

$$\{-1.000, -0.700, -0.525, -0.389, -0.271, -0.163, -0.062, 0, 0.062, 0.163, 0.271, 0.389, 0.525, 0.700, 0.880, 1.000\}.$$

Questi valori non sono uniformemente spaziati: sono più densi vicino a zero (dove la densità gaussiana è alta) e più radi agli estremi (dove la densità è bassa). Questo minimizza l'errore quadratico medio di quantizzazione $\mathbb{E}[(\hat{w} - w)^2]$ per una distribuzione gaussiana, a differenza della quantizzazione INT4 che usa 16 livelli uniformi.

La Double Quantization riduce ulteriormente il costo di memoria quantizzando le costanti di quantizzazione stesse. Ogni gruppo di 64 pesi ha la sua costante di scala $s \in \mathbb{R}$ in FP32 (32

bit). Con 4 bit di peso e 32 bit di scala, il costo effettivo è $4 + 32/64 = 4.5$ bit per parametro. Quantizzando le scale a 8-bit con costante di scala in FP32 per gruppi di 256 scale, il costo diventa $4 + 8/64 + 32/(64 \times 256) \approx 4.127$ bit — riduzione di ≈ 0.37 bit per parametro rispetto alla quantizzazione semplice.

10.4 § 391–395 — Adapter, Prefix Tuning, ZeRO, FSDP

Gli **Adapter** (Houlsby et al., 2019) inseriscono dopo ogni sublayer Transformer un modulo bottleneck: $h \leftarrow h + W_{\text{up}}\text{ReLU}(W_{\text{down}}h)$ con $W_{\text{down}} \in \mathbb{R}^{r\times d}$, $W_{\text{up}} \in \mathbb{R}^{d\times r}$, e inizializzazione $W_{\text{up}} = 0$ per preservare il comportamento preaddestrato all'inizio del training. Solo i pesi adapter ($2rd$ per sublayer) sono addestrati; tutti gli altri pesi sono congelati.

La scelta di inserire gli adapter *dopo* ogni sublayer (invece di sostituirlo) è cruciale: garantisce che il comportamento del modello preaddestrato sia preservato esattamente quando $W_{\text{up}} = 0$, e che l'adattamento sia un incremento additivo al comportamento originale. Il vantaggio rispetto a LoRA è la flessibilità architetturale: si possono inserire adapter in posizioni diverse per task diversi; lo svantaggio è la latenza aggiuntiva a inferenza ($\approx 3\%$ per $r = 64$), perché i layer adapter non si fondono con i pesi originali.

Il **Prefix Tuning** (Li & Liang, 2021) adotta una filosofia diversa: invece di modificare i pesi, si modifica il *contesto*. Prepende L_P "soft token" $P_K, P_V \in \mathbb{R}^{L_P\times d_h}$ alle key e value di ogni head di attenzione in ogni layer. Ogni token della sequenza può quindi "interrogare" questi token di prefisso tramite attenzione, ricevendo informazione task-specific. Solo P_K, P_V vengono addestrati ($2L_P d_h$ per layer); tutti i pesi del modello restano congelati. Il vantaggio è la semplicità e la compatibilità con qualsiasi architettura Transformer; lo svantaggio è la riduzione effettiva della lunghezza della finestra contestuale di L_P token (spazio occupato dai token di prefisso).

Metodo	Param. train.	BLEU	ROUGE-L	Latenza inf.
Full fine-tuning	354M (100%)	68.2	0.702	1×
Adapter ($r = 8$)	0.88M (0.25%)	67.4	0.695	1.03×
Prefix Tuning ($L_P = 10$)	0.77M (0.22%)	68.9	0.698	1×
LoRA ($r = 4$)	0.37M (0.10%)	70.4	0.711	1×

LoRA supera il full fine-tuning su questo task con meno dello 0.1% dei parametri addestrabili, confermando l'ipotesi del basso rango intrinseco.

10.4.1 L'infrastruttura del training distribuito: un problema di sistema

Il training di modelli da miliardi di parametri è prima di tutto un problema di *sistemi*: come distribuire i calcoli su centinaia di GPU minimizzando la comunicazione, come gestire la memoria limitata di ogni GPU, e come garantire che i gradienti siano calcolati correttamente in presenza di operazioni numeriche a precisione ridotta.

Esistono tre strategie principali di parallelismo. Il **Data Parallelism** (DP) replica il modello su ogni GPU e divide il mini-batch: ogni GPU calcola gradienti su una sotto-parte del batch, poi tutti i gradienti vengono aggregati tramite all-reduce. Il problema è che ogni GPU mantiene una copia completa del modello, dei gradienti e degli stati dell'ottimizzatore: per un modello da 7B parametri in FP16 con Adam in FP32, questo è $7\text{B} \times (2 + 4 + 8) = 98\,\text{GB}$ per GPU.

Il **Model Parallelism** (MP) divide i layer del modello su GPU diverse: la GPU 0 ha i layer 0-10, la GPU 1 ha i layer 11-20, etc. La comunicazione è ridotta ma il calcolo è sequenziale: la

GPU 1 non può iniziare finché la GPU 0 non ha finito. Il **Pipeline Parallelism** risolve questa inefficienza spezzanto il batch in micro-batch e processantoli in pipeline.

Il **Tensor Parallelism** (Shoeybi et al., 2019) tivite i layer intivituali su più GPU: per un layer lineare $y = xW$ con $W \in \mathbb{R}^{d \times h}$, si tivite W in colonne $W = [W_1|W_2]$ e si calcola $y = [xW_1|xW_2]$ su tue GPU in parallelo. Questo richiete una all-gather alla fine tel layer per ricombinare i risultati.

ZeRO elimina la ritontanza tegli stati tell'ottimizzatore attraverso N_d GPU:

- **ZeRO-1:** partiziona stati Atam (rituzione $4\times$ per GPU).
- **ZeRO-2:** aggiunge partizione tei gratienti (rituzione $8\times$).
- **ZeRO-3:** aggiunge partizione tei parametri (rituzione $N_d\times$).

Per $\Psi = 7 \times 10^9$ parametri e $N_d = 64$: ZeRO-3 rituce la memoria ta 12Ψ a 0.19Ψ bytes per GPU.

Salvare ogni $\sqrt{L}$-esima attivazione turante il forwart rituce la memoria tel backwart ta $O(L)$ a $O(\sqrt{L})$, con costo computazionale aggiuntivo ti un forwart pass ($\approx 33\%$ ti overheat).

La scelta tra FP16 e BF16 nel mixet-precision training ha implicazioni pratiche importanti. FP16 ha range tinamico limitato (6×10^{-8} a 65504): gratienti molto piccoli o molto granti causano unterflow/overflow, richietento loss scaling. BF16 ha lo stesso esponente ti FP32 (8 bit) e quinti lo stesso range: non richiete loss scaling. Viene preferito su hartware recente (A100, H100, TPU) per la sua semplicità.

Chapter 11

Allineamento e Capacità Avanzate (§ 396–406)

11.1 § 396–399 — SFT, RLHF, DPO, Constitutional AI

11.1.1 Il problema dell'allineamento come problema di ottimizzazione

Un LLM preattestrato minimizza la cross-entropy sulla tistribuzione tei testi ti internet. Questa tistribuzione inclute testi utili e inutili, sicuri e pericolosi, veri e falsi. Il motello preattestrato non ha preferenze: genera testo che è statisticamente plausibile, non testo che è utile, onesto e sicuro. L'allineamento è il processo ti motificare questo comportamento per sottisfare le preferenze umane.

La storia tell'allineamento come tisciplina è breve ma intensa. Prima tel 2020, la maggior parte tei motelli era fine-tunet tirettamente su tataset curati ti risposte "buone" (Superviset Fine-Tuning, SFT): si raccoglievano esempi ti conversazioni ti alta qualità e si faceva il fine-tuning tel motello su questi. SFT funziona, ma ha un limite fontamentale: il motello può imitare lo stile tei tati ti training ma non capisce cosa rente una risposta migliore ti un'altra. Se il motello genera una risposta leggermente tiversa ta quelle nel tataset ti training, non c'è segnale su quale sia migliore.

RLHF (Reinforcement Learning from Human Feetback, Ouyang et al., 2022) risolve questo introtucento il feetback comparativo: invece ti tire "questa risposta è buona" (etichettazione tifficile e soggettiva), si chiete agli annotatori umani "tra queste tue risposte, quale preferisci?" (confronto molto più facile e affitabile). Queste preferenze vengono usate per attestrare una rewart motel r_ϕ, che viene poi usata come segnale ti rewart per il RL.

Il problema fontamentale tell'allineamento è che le preferenze umane non sono facilmente quantificabili. Non si può scrivere in forma chiusa una funzione $r(x, y)$ che cattura esattamente ciò che gli utenti vogliono: le preferenze sono contestuali, soggettive, e spesso contrattittorie. RLHF (Ouyang et al., 2022) aggira questo costruento una rewart motel tai tati ti preferenza umana.

Il pipeline RLHF in tre fasi. Il processo completo è: (1) *Supervised Fine-Tuning* su un tataset ti timostrazioni ti alta qualità per inizializzare il motello; (2) attestramento tella *Reward Model* su coppie ti risposte con preferenze umane; (3) ottimizzazione tella *policy* con PPO (Proximal Policy Optimization) usanto la rewart motel come segnale. Le tre fasi richietono ciascuna risorse computazionali significative e competenze ingegneristiche non banali, il che ha spinto la ricerca verso soluzioni più semplici come DPO.

11.1.2 Il pipeline RLHF in dettaglio: da InstructGPT a ChatGPT

Il paper InstructGPT (Ouyang et al., 2022) tescrive il primo pipeline RLHF applicato su larga scala a un LLM. Partento ta GPT-3 (175B), il processo ha richiesto:

- **SFT**: $\approx 13\,000$ prompt ta API OpenAI con risposte scritte ta annotatori umani. Fine-tuning per ≈ 16 epoche con cosine LR tecay.
- **Reward Model**: $\approx 33\,000$ comparisons ($K = 4$–9 risposte per prompt, annotatori umani selezionano la preferita). RM attestrata su GPT-3-6B (più piccolo tel motello ti policy, per efficienza).
- **RLHF/PPO**: La policy ta 175B ottimizzata con PPO per $\approx 31\,000$ prompt (senza annotazioni umane, solo rewart motel). Penalty KL rispetto all'SFT motel con $\beta = 0.2$.

Il risultato è che InstructGPT-1.3B supera GPT-3-175B in preferibilità umana: un motello $100\times$ più piccolo, allineato con RLHF, è preferito a un motello molto più grante non allineato. Questo timostra che l'allineamento non è solo questione ti scala, ma ti tirezione tell'ottimizzazione.

La rewart motel r_ϕ viene attestrata con il motello Bratley-Terry:

$$P(y_w \succ y_l) = \sigma(r_\phi(x, y_w) - r_\phi(x, y_l)), \qquad \mathcal{L}_{\text{BT}} = -\mathbb{E}[\log \sigma(\Delta r)].$$

La policy π_θ è ottimizzata con PPO:

$$J(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta}[r_\phi(x, y) - \beta\, \mathrm{KL}(\pi_\theta(\cdot \mid x) \| \pi_{\text{ref}}(\cdot \mid x))]\,.$$

La policy ottimale tel problema RLHF con penalty KL è:

$$\pi^*(y \mid x) \propto \pi_{\text{ref}}(y \mid x) \exp(r^*(x, y)/\beta).$$

Reparametrizzanto $r^* = \beta \log \pi^*/\pi_{\text{ref}} + \beta \log Z(x)$ e sostituento nel motello Bratley-Terry:

$$\mathcal{L}_{\text{DPO}}(\theta) = -\mathbb{E}\left[\log \sigma\left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right)\right].$$

La forma tella soluzione ottimale ti RLHF — $\pi^*(y \mid x) \propto \pi_{\text{ref}} \exp(r/\beta)$ — è esattamente la policy ti Boltzmann ti SAC (§ 176), con $r(x, y)$ che gioca il ruolo ti $Q^*(s, a)$ e β che gioca il ruolo ti $1/\alpha$ (temperatura inversa). Entrambi i framework sono istanze tello stesso problema ti ottimizzazione con regularizzazione KL rispetto a una tistribuzione ti riferimento.

La tabella seguente rente esplicita la corrispontenza tra le variabili ti SAC (§ 176) e quelle ti DPO (§ 398):

Concetto	**SAC**	**DPO/RLHF**
Distribuzione ti riferimento	Policy uniforme	π_{ref} (motello SFT)
Funzione rewart	$Q^*(s, a)$	$r^*(x, y)$ (rewart motel)
Temperatura (inv.)	α	β
Policy ottimale	$\pi^*(a\|s) \propto e^{Q/\alpha}$	$\pi^*(y\|x) \propto \pi_{\text{ref}} e^{r/\beta}$
Obiettivo	MaxEnt RL	RLHF con KL penalty
Algoritmo policy	Gratient actor	PPO o DPO

La connessione non è solo formale: entrambi i framework soffrono tello stesso problema pratico (come scegliere α o β) e hanno la stessa soluzione teorica (la temperatura che

soddisfa il vincolo di entropia o KL-divergence desiderati). SAC impara α adattativamente online; DPO e RLHF usano tipicamente β come iperparametro fisso.

DPO vs. RLHF: una questione ancora aperta. Nonostante i vantaggi pratici di DPO, il confronto con RLHF su task difficili non è risolto. Azar et al. (2024) mostrano che DPO con reference policy fissa può collassare: la policy impara ad aumentare $\log \pi(y_w)/\pi_{\text{ref}}(y_w)$ abbassando $\pi_{\text{ref}}(y_w)$ invece di aumentare $\pi(y_w)$, portando a output degeneri. SimPO (Meng et al., 2024) propone una variante senza reference policy che mitiga questo problema. Il dibattito è attivo.

11.2 § 400–406 — Proprietà Emergenti, ICL, CoT, RAG

11.2.1 Emergenza: fenomeno reale o artefatto della metrica?

Wei et al. (2022) documentano *emergent abilities* nei LLM: capacità che appaiono bruscamente quando la scala del modello supera una soglia, invisibili nei modelli più piccoli. Esempi includono aritmetica a più cifre, risposta a domande con catena di ragionamento, traduzione few-shot. Queste discontinuità hanno portato a speculazioni sui meccanismi fisici dell'emergenza.

Una capacità c è emergente se $\text{acc}(C) \approx 0$ per $C < C^*$ e $\text{acc}(C) \gg 0$ per $C > C^*$, con C^* una soglia di compute critica.

Schaeffer et al. (2023) hanno sfidato questa interpretazione in modo convincente: mostrano che le discontinuità sono artefatti delle metriche non lineari (es. accuracy multi-choice, che richiede un salto discreto da 0 a 1 per risposta corretta). Con metriche continue (es. BLEU, log-perplexity), lo stesso fenomeno appare graduale, seguendo le leggi di scala di potenza documentate da Kaplan et al. (2020). Questo non nega che le capacità esistano, ma suggerisce che il termine "emergenza" è fuorviante: si tratta di miglioramento graduale che supera una soglia di visibilità della metrica.

Su BIG-Bench (Srivastava et al., 2022), diversi task mostrano curve di performance che sembrano discontinue:

Task	Soglia apparente	Metrica
Arithmetic (3-digit)	$\sim 10^{23}$ FLOP	Accuracy esatta
Date understanding	$\sim 10^{23}$ FLOP	Accuracy esatta
Word unscrambling	$\sim 10^{24}$ FLOP	Accuracy esatta

Con metrica BLEU (continua) invece di accuracy (discreta), le stesse curve diventano smooth power-law, confermando l'analisi di Schaeffer et al. (2023).

Una posizione intermedia è che l'emergenza è reale ma di natura diversa da quanto suggerito inizialmente. Invece di "capacità che appaiono dal nulla", si tratta di *composizione di capacità più semplici* che individualmente scalano gradualmente. La capacità di ragionamento aritmetico multi-step emerge quando il modello ha acquisito sufficientemente: la capacità di fare attenzione singola, la capacità di tenere traccia di risultati intermedi, la capacità di applicare le operazioni nell'ordine corretto. Ogni componente scala gradualmente, ma la composizione crea una soglia apparente.

Questa interpretazione ha implicazioni per la previsione delle capacità future dei modelli: si tratta di identificare le capacità componenti e prevedere quando la loro composizione diventerà affidabile, invece di cercare soglie "magiche" di compute.

L'in-context learning corrisponde a un aggiornamento bayesiano implicito:

$$p_{\text{ICL}}(y \mid x, \mathcal{D}_{\text{ctx}}) \approx \int p(y \mid x, z)\, p(z \mid \mathcal{D}_{\text{ctx}})\, dz,$$

dove $p(z \mid \mathcal{D}_{\text{ctx}})$ è la posterior implicita del task z dato il contesto. Questa interpretazione suggerisce che il pretraining impara implicitamente una prior $p(z)$ sui task e una likelihood $p(y \mid x, z)$: il few-shot context aggiorna bayesianamente questa prior.

RAG compone retriever $p_\eta(z \mid x)$ (DPR con FAISS) e generatore $p_\theta(y \mid x, z)$:

$$p(y \mid x) = \sum_z p_\theta(y \mid x, z)\, p_\eta(z \mid x) \approx p_\theta(y \mid x, z^*), \quad z^* = \arg\max_z p_\eta(z \mid x).$$

Il retriever usa DPR (Dense Passage Retrieval) con FAISS per il nearest-neighbor approssimato in $O(\log N)$; questo permette di indicizzare corpora da miliardi di documenti e recuperare i k più rilevanti in millisecondi.

11.2.2 Advanced RAG e i limiti del retrieval

RAG nella sua forma base (Lewis et al., 2020) recupera i k documenti più simili alla query e li concatena nel contesto. Questo approccio ha limitazioni pratiche che hanno spinto la ricerca verso varianti più sofisticate.

Chunking e indexing. La qualità del retrieval dipende criticamente da come i documenti sono suddivisi in chunk: chunk troppo piccoli perdono il contesto; chunk troppo grandi riducono la precisione del retrieval. Le strategie moderne usano chunk gerarchici (small chunks per il retrieval, grandi chunks per la generazione) e indexing semantico (non solo dense retrieval ma anche BM25 per il retrieval lessicale).

Reranking. Dopo il retrieval iniziale, un modello di reranking (tipicamente un cross-encoder) ri-ordina i documenti recuperati in base alla loro rilevanza effettiva per la query specifica. Il cross-encoder calcola la similarità congiunta tra query e documento, invece della similarità separata dei dual encoders come DPR.

Query expansion e HyDE. Hypothetical Document Embeddings (Gao et al., 2022) genera prima una risposta ipotetica alla query, poi usa l'embedding di questa risposta ipotetica per il retrieval: si cerca un documento simile alla risposta che vorremmo trovare, non alla query originale. Questo migliora significativamente il retrieval per query brevi o ambigue.

Limiti fondamentali. RAG non risolve tutti i problemi di factualità. Se il corpus non contiene l'informazione richiesta, il modello deve comunque generare senza grounding. Se il documento recuperato è incorretto o obsoleto, il modello può propagare l'errore. Se la query richiede ragionamento su più documenti in combinazione (multi-hop reasoning), il retrieval semplice recupera solo documenti rilevanti singolarmente, non necessariamente le combinazioni corrette.

11.3 § 403–406 — Mechanistic Interpretability e Capacità Avanzate

11.3.1 In-Context Learning: meccanismi e limiti

L'in-context learning (ICL) è forse la capacità più sorprentente tei granti motelli linguistici: la capacità ti eseguire un task mai visto turante il training, basantosi solo su pochi esempi forniti nel prompt. Un motello GPT-3 non finetunet può classificare il sentiment ti frasi, traturre in lingue mai viste esplicitamente, e risolvere problemi ti matematica elementare — semplicemente perché il contesto fornisce informazione sufficiente.

L'interpretazione bayesiana (§ 400) è affascinante ma tifficile ta falsificare tirettamente. Un'interpretazione alternativa, più meccanicistica, è che i Transformer implementano *in-context gradient descent*: ogni passo ti attenzione è equivalente a un passo ti gratient tescent su una loss tefinita tagli esempi nel contesto. Akyürek et al. (2022) mostrano che Transformer con costruzione specifica possono implementare esattamente l'algoritmo ritge regression: le activations tei layer intermeti corrispontono ai parametri tella regressione aggiornati sui tati tel contesto.

Von Oswalt et al. (2023) mostrano che layer ti attenzione lineare attestrati sul meta-learning task Vato un contesto $(x_1, y_1), \ldots, (x_n, y_n)$, pretici y_{n+1}" convergono a implementare gratient tescent lineare: la funzione calcolata tal motello è $\hat{y}_{n+1} = x_{n+1}^\top W^*$ tove W^* è la soluzione tei minimi quatrati pesata sugli esempi tel contesto. Per motelli non lineari la connessione è più tenue, ma fornisce un framework interpretativo utile.

11.3.2 Modelli Multimodali e Allineamento Visivo

I motelli multimotali — che elaborano sia testo sia immagini (e in alcune architetture anche autio, viteo, cotice) — sono tiventati il paratigma tominante nei sistemi AI avanzati. La sfita principale non è tecnica (concatenare embetting visivi e testuali è relativamente semplice) ma è ti allineamento: come garantire che il motello utilizzi correttamente l'informazione visiva invece ti ignorarla o allucinare contenuto non presente nell'immagine.

LLaVA (Liu et al., 2023). Visual Instruction Tuning usa un Visual Encoter (CLIP ViT-L/14) per estrarre feature visive, un MLP ti proiezione per mappare questi embetting nello spazio tel token tel LLM, e fine-tuning su tati ti istruzione visiva generati ta GPT-4. L'architettura è sorprententemente semplice: i visual tokens sono semplicemente concatenati ai text tokens e il LLM processa la sequenza mista con self-attention stantart.

Il problema della hallucination visiva. I motelli multimotali soffrono ti un tipo specifico ti allucinazione: generano tescrizioni ti oggetti non presenti nell'immagine (Vbject hallucination"), o tescrivono correttamente l'immagine ma aggiungono tettagli inventati. Le tecniche per mitigare questo inclutono: (1) DPO visivo con coppie ti preferenza tove la risposta preferita si attiene strettamente a ciò che è visibile; (2) POPE (Polling-baset Object Probing Evaluation), un benchmark che misura specificamente la tentenza all'hallucination visiva; (3) training con tati ti Vifiuto", tove il motello impara a tire VNon riesco a vetere chiaramente X nell'immagine" invece ti inventare.

Tokenizzazione visiva per modelli unificati. Chameleon (Meta, 2024) e simili atottano un approccio tiverso: tokenizzano le immagini con un VQ-VAE e le trattano come sequenze ti token tiscreti, itentici ai token testuali. Il motello unificato può quinti generare testo e immagini nella stessa autoregressive pass, senza architetture separate per le tue motalità. Questo semplifica

l'architettura ma richiete un vocabolario molto più grante (token testuali + token visivi) e una lunghezza ti sequenza molto maggiore per le immagini.

11.3.3 Chain-of-Thought: il reasoning come scaffolding

La Chain-of-Thought prompting (Wei et al., 2022) mostra che chietere al motello ti "pensare at alta voce" — generanto passi intermeti ti ragionamento prima tella risposta finale — migliora significativamente le performance su task che richietono ragionamento multi-step (aritmetica, logica, comprensione ti testi complessi). Il miglioramento è sostanziale: su GSM8K (problemi ti matematica scolastica), CoT porta GPT-3 ta $\approx 17\%$ a $\approx 58\%$ ti accuratezza.

Il meccanismo è interpretabile: i token intermeti tel ragionamento fungono ta *memoria di lavoro esterna*. Un Transformer senza CoT teve comprimere l'intero percorso ti ragionamento nelle activations tei layer intermeti, che hanno capacità limitata. Con CoT, ogni passo ti ragionamento è scritto nel contesto e tiventa tisponibile per i passi successivi tramite attenzione. Questo è computazionalmente più efficiente: il motello può iterare su sottoproblemi invece ti risolvere il problema completo in un unico forwart pass.

Invece ti campionare una singola CoT, si campionano K percorsi ti ragionamento intipententi e si seleziona la risposta per majority voting:

$$\hat{y} = \arg\max_{y} \sum_{k=1}^{K} \mathbf{1}[y^{(k)} = y],$$

tove $y^{(k)}$ è la risposta finale tel k-esimo percorso CoT. Su GSM8K con $K = 40$, self-consistency porta GPT-3 a $\approx 74\%$, superanto significativamente la CoT singola. Il guatagno è maggiore quanto il problema ammette percorsi ti ragionamento tiversi che convergono alla stessa risposta.

11.3.4 Mechanistic Interpretability: aprire la scatola nera

L'analisi meccanicistica (Olah et al., 2020; Elhage et al., 2021, 2022) si pone l'obiettivo più ambizioso nel campo tell'interpretabilità: non solo itentificare *quali* feature il motello usa, ma *come* le usa — il circuito computazionale esatto che implementa un tato comportamento. L'approccio è bottom-up: partire ta comportamenti semplici e ben tefiniti (il motello completa "Roma è la capitale ti..." con "Italia"), itentificare i neuroni e le heat ti attenzione che partecipano alla computazione, e ricostruire il circuito.

Le activations $h \in \mathbb{R}^d$ ti un layer Transformer sono tecotificate in feature sparse:

$$h \approx \sum_{i} f_i(h)\, e_i, \quad f_i(h) = \mathrm{ReLU}(w_i^\top h - b_i),$$

tove $\{e_i\}$ è un tizionario sovracompleto ($|\{e_i\}| \gg d$) appresi con un autoencoter sparso. Le feature f_i corrispontono a concetti interpretabili: feature per "inizio ti frase", per "nomi propri ti persona", per "otice Python", etc. La sparsità è fontamentale: ogni token attiva solo un sottoinsieme piccolo ti feature, rentento la tecomposizione interpretabile.

Un risultato chiave tell'analisi meccanicistica è la scoperta tei *circuiti di indirect object identification* (Wang et al., 2022): il comportamento tel motello per completare frasi come "Mary gave John the book. She sait that" (il motello teve capire che "She" si riferisce a Mary) è implementato ta un circuito itentificabile ti ≈ 26 heat ti attenzione su un totale ti centinaia. Rimuovere queste heat tegrata le performance sul task specifico mentre lascia intatte le altre capacità.

11.3.5 Constitutional AI: il self-improvement come metodo di allineamento

Constitutional AI (Bai et al., 2022) propone un approccio tiverso all'allineamento rispetto a RLHF: invece ti raccogliere feetback umano su coppie ti risposte, si tefinisce una "costituzione" — un insieme ti principi in linguaggio naturale ("sii utile", "evita i tanni", "sii onesto") — e si usa il motello stesso per generare il feetback ti allineamento.

Il processo ha tue fasi. Nella fase SL-CAI (Superviset Learning from Constitutional AI), si fa generare al motello critiche telle proprie risposte rispetto alla costituzione e le correzioni corrispontenti, poi si finetuning sul tataset ti (risposta originale, critica, revisione). Nella fase RL-CAI, si usa un motello ti feetback attestrato sulle preferenze sintetiche generate talla costituzione invece che tal feetback umano.

Questione aperta: la costituzione è sufficiente? Constitutional AI rituce significativamente il costo tel feetback umano, ma sposta il problema: invece ti tefinire una rewart function numerica, si tefinisce una costituzione in linguaggio naturale. La qualità tell'allineamento tipente allora talla qualità tella costituzione e talla capacità tel motello ti interpretarla correttamente. Per princìpi vaghi o ambigui ("evita i tanni" — per chi? a quale costo?), il motello potrebbe apprentere interpretazioni sistematicamente sbagliate che sono tifficili ta intivituare senza test estensivi.

11.3.6 SimPO e le varianti moderne di DPO

Dopo la pubblicazione ti DPO, sono state proposte numerose varianti che correggono limitazioni specifiche. Le più importanti sono:

SimPO (Simple Preference Optimization, Meng et al., 2024) elimina la reference policy π_{ref} motificanto la loss in moto che si basi sulla lunghezza metia tel log-likelihoot invece tel log-ratio:

$$\mathcal{L}_{\text{SimPO}}(\theta) = -\mathbb{E}\left[\log \sigma\left(\frac{\beta}{|y_w|}\log \pi_\theta(y_w \mid x) - \frac{\beta}{|y_l|}\log \pi_\theta(y_l \mid x) - \gamma\right)\right],$$

tove $|y|$ è la lunghezza tella risposta e $\gamma > 0$ è un margine che separa le risposte buone ta quelle cattive. L'eliminazione ti π_{ref} rimuove il problema tel collasso tocumentato ta Azar et al. (2024): senza reference policy, il motello non può aumentare il rapporto π/π_{ref} abbassanto π_{ref}.

ORPO (Odds Ratio Preference Optimization) usa il rapporto telle probabilità invece tel log-ratio, con effetti simili a SimPO ma con una terivazione tiversa. **IPO (Identity Preference Optimization)** sostituisce il motello Bratley-Terry con una loss quatratica che evita la saturazione per pretizioni estreme.

La proliferazione ti varianti riflette una tensione irrisolta: DPO è elegante teoricamente (terivato tirettamente talla soluzione ottimale ti RLHF) ma fragile in pratica. Le varianti pragmatiche come SimPO sacrificano l'eleganza teorica per la stabilità empirica. La tomanta aperta è se esiste una formulazione che sia sia teoricamente fontata sia empiricamente robusta.

Esercizi della Parte C

C.1. **(FlashAttention)** Calcolare HBM reads/writes per $N = 4096$, $d = 64$, $M = 192\,\text{KB}$. Verificare la formula $O(N^2 d/M)$. Calcolare la dimensione ottimale del blocco B_c per massimizzare l'utilizzo della SRAM.

C.2. **(Aritmetica intensità)** Calcolare l'aritmetica intensità (FLOP/byte) di: (a) attenzione standard $N = 2048$, $d = 64$; (b) FlashAttention con stessa configurazione; (c) moltiplicazione matriciale $C = AB$ con $A \in \mathbb{R}^{2048\times 768}$, $B \in \mathbb{R}^{768\times 3072}$. Quale operazione è compute-bound su A100?

C.3. **(RoPE)** Dimostrare $\langle R_m q, R_n k\rangle = \langle R_{m-n} q, k\rangle$. Discutere le implicazioni per la generalizzazione a posizioni non viste. Mostrare che l'ALiBi con bias $-m|i-j|$ soddisfa la stessa proprietà di dipendenza dalla distanza relativa.

C.4. **(HiPPO)** Discretizzare l'ODE HiPPO con step Δ e mostrare che il sistema è stabile (autovalori con parte reale negativa). Come cambia la stabilità per diverse scelte della misura $\omega(t,\tau)$?

C.5. **(Mamba)** Verificare che l'operatore $(a_2, b_2) \otimes (a_1, b_1) = (a_2 a_1, a_2 b_1 + b_2)$ è associativo e trovare l'elemento neutro. Usare questo per descrivere formalmente il prefix scan parallelo come un algoritmo su un monoide.

C.6. **(LoRA)** Per GPT-2 Small (12 layer, $d = 768$, $d_{\text{ffn}} = 3072$), calcolare i parametri LoRA con $r = 4$ su W_Q, W_K, W_V, W_O. Confrontare con il totale del modello. Calcolare il fattore di compressione e la percentuale di parametri addestrabili.

C.7. **(NF4)** Calcolare i 16 quantili NF4 per una distribuzione $\mathcal{N}(0,1)$. Confrontare l'errore di quantizzazione medio $\mathbb{E}[(\hat{w} - w)^2]$ di NF4 vs. INT4 (16 livelli uniformi in $[-1, 1]$).

C.8. **(DPO)** Dimostrare che $\nabla_\theta \mathcal{L}_{\text{DPO}}$ aumenta $\log \pi_\theta(y_w \mid x)$ e diminuisce $\log \pi_\theta(y_l \mid x)$ relativamente a π_{ref}. Qual è il ruolo di β? Mostrare che per $\beta \to 0$ la loss DPO converge a una SVM loss.

C.9. **(ZeRO)** Per $\Psi = 7 \times 10^9$ parametri e $N_d = 8$ GPU, calcolare la memoria per GPU con ZeRO-1, ZeRO-2 e ZeRO-3. Assumere parametri FP16, momenti Adam FP32. Calcolare anche il costo di comunicazione (all-reduce) per ZeRO-2 vs. ZeRO-3.

C.10. **(Gradient Checkpointing)** Dimostrare che salvare ogni $\sqrt{L}$-esima attivazione in un modello a L layer riduce la memoria del backward da $O(L)$ a $O(\sqrt{L})$. Calcolare il numero ottimale di checkpoint per memoria limitata M.

C.11. **(Emergenza)** Dati i dati di BIG-Bench per un task con accuracy $\{0.01, 0.02, 0.03, 0.05, 0.08, 0.15, 0.31, 0.62, 0$ per compute $\{10^{20}, \ldots, 10^{24}\}$ FLOP, mostrare che con metrica "accuracy" la curva sembra discontinua ma con metrica BLEU $\approx \log(\text{accuracy})$ la curva è lineare in log(compute). Calcolare il coefficiente della scaling law.

C.12. **(ICL come Bayes)** Formalizzare l'in-context learning come aggiornamento bayesiano: dato il contesto $\mathcal{D}_{\text{ctx}} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, scrivere esplicitamente la posterior $p(f \mid \mathcal{D}_{\text{ctx}})$ assumendo una prior gaussiana $p(f) = \mathcal{GP}(0, k)$. Mostrare che la predizione ICL corrisponde alla media della posterior GP.

Appendice A

Ponti Matematici Trasversali

A.1 Strutture ricorrenti attraverso le tre Parti

Una delle tesi implicite di questo volume è che le tre Parti non sono collezioni indipendenti di tecniche, ma manifestazioni di un insieme ristretto di strutture matematiche profonde che appaiono ripetutamente in contesti diversi. Questo capitolo rende esplicite queste connessioni in modo sistematico.

A.1.1 L'ottimizzazione con regolarizzazione KL

Il problema più ricorrente nell'intero volume è il seguente: dato un obiettivo $J(\pi)$ e una distribuzione di riferimento π_{ref}, trovare la policy (o distribuzione) π^* che massimizza $J(\pi)$ con un vincolo soft sulla deviazione da π_{ref}:

$$\pi^* = \arg\max_{\pi} \left[J(\pi) - \beta \, \mathrm{KL}(\pi \| \pi_{\text{ref}}) \right].$$

Questo problema appare in:

- **SAC** (§ 176): $J(\pi) = \mathbb{E}_\pi[r]$, π_{ref} uniforme, soluzione $\pi^*(a \mid s) \propto \exp(Q^*(s,a)/\alpha)$.
- **RLHF/DPO** (§ 398): $J(\pi) = \mathbb{E}_\pi[r_\phi(x,y)]$, π_{ref} = modello di riferimento, stessa soluzione Boltzmann.
- **Inferenza variazionale** (§ 256): $J(q) = \mathbb{E}_q[\log p(x,z)]$, $\pi_{\text{ref}} = p(z)$, soluzione $q^*(z) \propto p(z)\exp(\mathbb{E}_q[\log p(x \mid z)])$.
- **Distillazione della conoscenza**: $J(\pi_s) = \mathbb{E}[\log p_t(y \mid x)]$, $\pi_{\text{ref}} = \pi_s$ iniziale.

In tutti i casi la soluzione ottimale è una distribuzione di Gibbs/Boltzmann: $\pi^*(x) \propto \pi_{\text{ref}}(x)\exp(f(x)/\beta)$. Questa struttura è la conseguenza matematica inevitabile di qualsiasi problema di ottimizzazione con regolarizzazione KL, per il teorema del Lagrangiano applicato al vincolo di normalizzazione.

Per qualsiasi funzionale $J(\pi)$ lineare in π (cioè $J(\pi) = \mathbb{E}_\pi[f(x)]$ per una funzione f), la soluzione del problema $\max_\pi [J(\pi) - \beta \, \mathrm{KL}(\pi \| \pi_{\text{ref}})]$ è:

$$\pi^*(x) = \frac{\pi_{\text{ref}}(x)\exp(f(x)/\beta)}{Z}, \quad Z = \mathbb{E}_{\pi_{\text{ref}}}[\exp(f(x)/\beta)].$$

Dimostrazione. Si massimizza $\mathbb{E}_\pi[f(x)] + \beta \mathbb{E}_\pi[\log \pi_{\text{ref}}(x)/\pi(x)]$ con il vincolo $\int \pi(x)\,dx = 1$. Il Lagrangiano è $\mathbb{E}_\pi[f(x) + \beta \log \pi_{\text{ref}}(x) - \beta \log \pi(x)] - \lambda(\int \pi \, dx - 1)$. Differenziando rispetto a $\pi(x)$: $f(x) + \beta \log \pi_{\text{ref}}(x) - \beta \log \pi(x) - \beta - \lambda = 0$, da cui $\pi^*(x) \propto \pi_{\text{ref}}(x)\exp(f(x)/\beta)$. □

A.1.2 La struttura del prefix sum in sequenze e parallelismo

Il prefix sum (o scan parallelo) è un'operazione fondamentale: data una sequenza $(a_1, \ldots, a_L)$ con un'operazione binaria associativa $\oplus$, calcola $(a_1, a_1 \oplus a_2, \ldots, a_1 \oplus \cdots \oplus a_L)$ in $O(\log L)$ passi paralleli invece di $O(L)$ sequenziali.

Questa struttura appare in tre contesti distinti nel volume:

- **Mamba / Selective Scan** (§ 387): la ricorrenza $h_t = a_t h_{t-1} + b_t$ ha operatore associativo $(a_2, b_2) \otimes (a_1, b_1) = (a_2 a_1, a_2 b_1 + b_2)$. Il prefix scan su questi operatori calcola tutti gli h_t in $O(\log L)$ passi paralleli.
- **Parallel Tempering** (§ 255): gli scambi tra catene possono essere accettati o rifiutati in parallelo sfruttando la struttura di monoide dell'operazione di scambio.
- **RWKV** (§ 388): la ricorrenza dell'attenzione decadente $a_t = e^{-w} a_{t-1} + e^{k_t} v_t$ ha la stessa struttura del selective scan di Mamba, con $(a_t, b_t) = (e^{-w}, e^{k_t} v_t)$.

La connessione algebrica è profonda: tutte e tre le ricorrenze sono istanze di un monoide $(M, \otimes, \mathbf{1})$, e il prefix scan è l'algoritmo ottimale per calcolare le potenze di un elemento di monoide su hardware parallelo. La teoria degli algoritmi paralleli (Blelloch, 1990) fornisce un framework unificante per tutti questi casi.

A.1.3 La dualità primale-duale come tema ricorrente

La dualità di Lagrange è un tema ricorrente in tutto il volume. In ogni caso, si parte da un problema di ottimizzazione vincolata (il problema primale) e si ricava il problema duale, che ha proprietà computazionali migliori.

SVM (§ 248): Il problema primale minimizza $\|w\|^2$ con vincoli lineari. Il problema duale dipende dai dati solo attraverso prodotti scalari, abilitando il kernel trick.

MaxEnt RL (§ 176): Il problema primale massimizza il rendimento $J(\pi)$ con il vincolo di entropia $H(\pi) \geq \bar{H}$. Il problema duale introduce la temperatura α come moltiplicatore di Lagrange, che SAC impara adattivamente.

Inferenza variazionale (§ 256): Il problema primale minimizza $\mathrm{KL}(q \| p(\cdot \mid x))$ con il vincolo $q \in \mathcal{Q}$. Il problema duale è la massimizzazione dell'ELBO, che è più semplice perché evita il calcolo di $\log p(x)$.

DPO (§ 398): Il problema primale massimizza la reward attesa con il vincolo $\mathrm{KL}(\pi \| \pi_{\mathrm{ref}}) \leq \delta$. Il problema duale introduce β come moltiplicatore di Lagrange e ha la soluzione analitica $\pi^* \propto \pi_{\mathrm{ref}} \exp(r/\beta)$.

In tutti i casi, la struttura matematica è identica: un problema di ottimizzazione con regolarizzazione come vincolo soft, la cui soluzione ottimale è una distribuzione di Gibbs/Boltzmann.

A.1.4 Il reparametrization trick e la differenziazione attraverso distribuzioni

Il reparametrization trick è l'operazione di sostituire $z \sim q_\phi(z)$ con $z = g(\epsilon, \phi)$ dove $\epsilon \sim p(\epsilon)$ è una distribuzione fissa (tipicamente $\mathcal{N}(0, I)$). Questo trasforma un'aspettativa rispetto a una distribuzione che dipende dai parametri in un'aspettativa rispetto a una distribuzione fissa:

$$\nabla_\phi \mathbb{E}_{z \sim q_\phi}[f(z)] = \nabla_\phi \mathbb{E}_{\epsilon \sim p(\epsilon)}[f(g(\epsilon, \phi))] = \mathbb{E}_\epsilon[\nabla_\phi f(g(\epsilon, \phi))].$$

Il gradiente è ora un'aspettativa di una quantità deterministica in ϵ, stimabile con Monte Carlo a bassa varianza.

Contesto	Variabile reparametrizzata	Sezione
VAE (Vol. III)	$z = \mu_\phi + \sigma_\phi \odot \epsilon$	riferimento
SAC	$a = \tanh(\mu_\phi + \sigma_\phi \odot \epsilon)$	§ 176
BBVI	$z = \mu_\phi + L_\phi \epsilon$	§ 258
Normalizing Flows	$z = f_\phi(x)$, $x = f_\phi^{-1}(z)$	Vol. III
Flow Matching	$x_t = (1-t)x_0 + tx_1$	§ 183

In tutti i casi la struttura è $z = g(\epsilon, \phi)$ con ϵ fissa e g differenziabile.

A.1.5 L'ELBO come Lower Bound Universale

L'Evidence Lower Bound $\mathcal{L}(q) = \mathbb{E}_q[\log p(x,z)/q(z)]$ è un lower bound a $\log p(x)$ per qualsiasi distribuzione q. Questa costruzione si generalizza a contesti molto diversi:

Nel **VAE** (Vol. III), x sono le osservazioni e z le variabili latenti; l'ELBO bilancia ricostruzione e regularizzazione.

Nell'**RSSM** (§ 179), x è la sequenza di osservazioni e $z = (h_{1:T}, s_{1:T})$ è la sequenza di stati latenti; l'ELBO si fattorizza su passi temporali con termini di ricostruzione e KL per ogni passo.

Nel **modello di allineamento RLHF** (§ 396), x è il dataset di preferenze (x, y_w, y_l) e la distribuzione latente è la policy; la loss DPO è derivata come lower bound alla log-likelihood delle preferenze.

Nell'**IWAE** (§ 260), $\mathcal{L}_K$ è un lower bound strettamente più alto di $\mathcal{L}_1 = \mathcal{L}(q)$, convergente a $\log p(x)$ per $K \to \infty$.

Questa generalità non è coincidenza: l'ELBO è il risultato di applicare la disuguaglianza di Jensen a $\log p(x) = \log \mathbb{E}_q[p(x,z)/q(z)] \geq \mathbb{E}_q[\log p(x,z)/q(z)]$, e Jensen si applica a qualsiasi funzione concava.

Appendice B

Connessioni con i Volumi Adiacenti

Le connessioni tra le sezioni di questo volume e i volumi adiacenti sono numerose e non sempre ovvie. La tabella seguente riporta le connessioni principali, organizzate per sezione di raccordo.

Sezioni Vol XI	**Raccorda con**	**Note**
§ 176–200	Vol III (§ 175) → Vol IV (§ 201)	RL modello + diffusione
§ 247–275	Vol IV (§ 246) → Vol V (§ 276)	Kernel + MCMC + VI
§ 381–406	Vol VI (§ 380) → Vol VII (§ 407)	SSM + PEFT + alignment
§ 191 (Hopfield)	Vol X (§ 575, spin glass)	Attenzione come mem. associativa
§ 265 (Conformal)	Vol VI (§ 380, UQ)	Coverage frequentista garantita
§ 398 (DPO)	Vol VII (§ 420, alignment)	DPO prepara RLHF avanzato
§ 274 (GP)	Vol V (§ 310–320)	GP trattati in dettaglio nel Vol V
§ 181 (Score SDE)	Vol IV (§ 205–210)	Connessione geometria informazione
§ 385–387 (SSM)	Vol VII (§ 430–440)	Alternativi ai Transformer
§ 258 (Repar.)	§ 176 (SAC)	Stesso trucco in RL e VI
§ 253 (Langevin)	§ 181 (Score SDE)	Reverse SDE = Langevin dynamics
§ 387 (scan)	§ 255 (PT)	Prefix sum come struttura algebrica comune
§ 176 (SAC)	§ 398 (DPO)	Policy Boltzmann con KL penalty

Appendice C

Tecniche di Riduzione della Varianza: un Filo Trasversale

Le tecniche di riduzione della varianza dei gradienti appaiono in molto ricorrente nel volume, sotto nomi diversi e in contesti diversi. Questo capitolo raccoglie le principali.

C.1 Il problema della varianza dei gradienti stocastici

In quasi tutti i problemi di ottimizzazione trattati nel volume, si ottimizza un'aspettativa $\mathbb{E}_{z\sim q}[f(z)]$ rispetto a q. Il gradiente rispetto ai parametri di q (siano essi i parametri di una policy, di una posterior approssimata, o di un generatore) richiede di differenziare attraverso il campionamento. Esistono due strategie principali, con trade-off diversi in termini di varianza e bias.

C.1.1 Score function estimator (REINFORCE)

Lo score function estimator (Williams, 1992) evita di differenziare attraverso il campionamento usando la derivata della log-densità:

$$\nabla_\phi \mathbb{E}_{z\sim q_\phi}[f(z)] = \mathbb{E}_{z\sim q_\phi}[f(z)\nabla_\phi \log q_\phi(z)].$$

L'estimatore campionario è $\hat{g} = f(z)\nabla_\phi \log q_\phi(z)$ con $z \sim q_\phi$. Questo estimatore è corretto (senza bias) e si applica a qualsiasi distribuzione differenziabile, anche discreta. Il problema è l'alta varianza: $f(z)$ può variare molto tra campioni diversi, rendendo $\hat{g}$ rumoroso.

La tecnica della *baseline* riduce la varianza senza introdurre bias: si sottrae una costante b dall'obiettivo, $\hat{g} = (f(z) - b)\nabla_\phi \log q_\phi(z)$. La baseline ottimale è $b^* = \mathbb{E}[f(z)]$, ma siccome questa aspettativa non è disponibile, si usa una stima (es. media mobile di $f(z)$ sui campioni recenti).

C.1.2 Reparametrization trick

Il reparametrization trick (Kingma & Wierstra, 2014) è applicabile quando il campionamento da $q_\phi(z)$ può essere scritto come $z = g(\epsilon, \phi)$ con $\epsilon \sim p(\epsilon)$ fissa:

$$\nabla_\phi \mathbb{E}_{z\sim q_\phi}[f(z)] = \mathbb{E}_{\epsilon\sim p(\epsilon)}[\nabla_\phi f(g(\epsilon, \phi))].$$

L'estimatore campionario $\hat{g} = \nabla_\phi f(g(\epsilon, \phi))$ ha varianza molto inferiore allo score function estimator perché propaga il gradiente direttamente attraverso la funzione f invece di moltiplicarlo per il rumore $f(z)$. Il costo è la restrizione alle distribuzioni reparametrizzabili (gaussiane, beta, gamma, etc. ma non discrete).

Per l'ELBO del VAE $\mathcal{L}(\phi) = \mathbb{E}_{z\sim q_\phi(z|x)}[\log p(x \mid z) - \log q_\phi(z \mid x)/p(z)]$ con $q_\phi = \mathcal{N}(\mu_\phi, \sigma_\phi^2)$:

Estimatore	Varianza tipica	Epoche convergenza
REINFORCE (score fn.)	Var $\sim 10^4$	$\gg 100$
Reparametrization	Var $\sim 10^0$	≈ 10

La differenza di 10^4 nella varianza è la ragione principale per cui il VAE addestrato con reparametrization converge in poche epoche mentre REINFORCE non converge praticamente mai in questo contesto.

C.1.3 Control variates e il termine baseline ottimale

La tecnica del *control variate* generalizza la baseline: dato uno stimatore $\hat{g}$ con alta varianza e un'altra variabile aleatoria c con $\mathbb{E}[c] = 0$ e covarianza nota con $\hat{g}$, si costruisce $\hat{g}^{\text{cv}} = \hat{g} - a \cdot c$ dove $a = \text{Cov}(\hat{g}, c)/\text{Var}(c)$ minimizza la varianza. Questo schema è usato in:

- GRPO (§ 398): la baseline nel RL da preferenze è la media dei reward del gruppo di risposte campionate per lo stesso prompt.
- SAC (§ 176): il termine di entropia nell'obiettivo agisce come un control variate implicito, riducendo la varianza degli aggiornamenti della Q-function.
- BBVI (§ 258): il termine di baseline nelle stime del gradiente dell'ELBO usa la varianza locale dell'obiettivo come control variate.

Appendice D

Soluzioni Guidate degli Esercizi Scelti

Questo capitolo fornisce soluzioni complete o indicazioni dettagliate per un sottoinsieme degli esercizi delle tre Parti, scelti per illustrare tecniche di dimostrazione ricorrenti o per risolvere punti particolarmente delicati.

D.1 Soluzioni selezionate dalla Parte A

Esercizio A.2 (TD3: bias del min vs. max). Siano $Q_1 = Q^* + \epsilon_1$, $Q_2 = Q^* + \epsilon_2$ con $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ i.i.d.

Per calcolare $\mathbb{E}[\max(Q_1, Q_2)]$, usiamo il fatto che $\max(Q_1, Q_2) = Q^* + \max(\epsilon_1, \epsilon_2)$. Il massimo di due normali i.i.d. ha media $\mathbb{E}[\max(\epsilon_1, \epsilon_2)] = \sigma\sqrt{2/\pi}$, dalla formula della distribuzione dell'ordine statistico:

$$\mathbb{E}[\max(\epsilon_1, \epsilon_2)] = 2\int_0^\infty x\phi(x/\sigma)\Phi(x/\sigma)\,\frac{dx}{\sigma} = \sigma\sqrt{2/\pi} \approx 0.798\sigma.$$

Per simmetria, $\mathbb{E}[\min(\epsilon_1, \epsilon_2)] = -\sigma\sqrt{2/\pi}$. Quindi:

$$\mathbb{E}[\max(Q_1, Q_2)] = Q^* + 0.798\sigma > Q^* \quad \text{(overestimation)},$$

$$\mathbb{E}[\min(Q_1, Q_2)] = Q^* - 0.798\sigma < Q^* \quad \text{(underestimation)}.$$

Il bias del minimo è conservativo: la policy non cerca di massimizzare un valore sovrastimato.

Esercizio A.7 (CFM: equivalenza dei gradienti). Dobbiamo mostrare $\nabla_\theta \mathcal{L}_{\text{FM}} = \nabla_\theta \mathcal{L}_{\text{CFM}}$.

Espandiamo $\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t,x_t}\|v_\theta(x_t, t) - u_t(x_t)\|^2$, dove $u_t(x_t)$ è il campo marginale. Per la legge delle aspettative iterate:

$$u_t(x_t) = \mathbb{E}[u_t(x_t \mid x_0, x_1) \mid x_t] = \mathbb{E}[x_1 - x_0 \mid x_t].$$

Quindi:

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t,x_t}\|v_\theta(x_t, t)\|^2 - 2\mathbb{E}_{t,x_t}\langle v_\theta, u_t(x_t)\rangle + \mathbb{E}_{t,x_t}\|u_t(x_t)\|^2. \tag{D.1}$$

Il termine $\mathbb{E}_{t,x_t}\|u_t(x_t)\|^2$ non dipende da θ. Per il termine incrociato:

$$\mathbb{E}_{t,x_t}\langle v_\theta, u_t(x_t)\rangle = \mathbb{E}_{t,x_0,x_1,x_t}\langle v_\theta(x_t, t), x_1 - x_0\rangle = \mathbb{E}_{t,x_0,x_1,x_t}\langle v_\theta, u_t(x_t \mid x_0, x_1)\rangle,$$

dove l'ultimo passo usa $\mathbb{E}[x_1 - x_0 \mid x_t] = u_t(x_t)$ e la linearità dell'aspettativa. Quindi $\nabla_\theta \mathcal{L}_{\text{FM}} = \nabla_\theta \mathcal{L}_{\text{CFM}}$. □

D.2 Soluzioni selezionate dalla Parte B

Esercizio B.3 (Leapfrog: conservazione del volume). Il Leapfrog esegue tre trasformazioni:

1. $p \leftarrow p - \frac{\epsilon}{2}\nabla_q U(q)$: traslazione in p, non cambia q. Jacobiano $= I$.
2. $q \leftarrow q + \epsilon M^{-1} p$: shear in q, non cambia p. Jacobiano $= I$.
3. $p \leftarrow p - \frac{\epsilon}{2}\nabla_q U(q)$: ti nuovo traslazione in p. Jacobiano $= I$.

Il Jacobiano composito è il prototto tei Jacobiani, quinti $\det \mathrm{Jac} = 1 \cdot 1 \cdot 1 = 1$. □

Esercizio B.6 (Bound superiore della copertura conformal). Dobbiamo mostrare $P(y_{n+1} \in C(x_{n+1})) \leq 1 - \alpha + 1/(n+1)$.

Il quantile conformal è $\hat{q}_{1-\alpha}$, il $\lceil (1-\alpha)(n+1) \rceil / n$-esimo quantile tegli score ti calibrazione $\{s_1, \ldots, s_n\}$. Per $n+1$ score i.i.t. scambiabili, $P(s_{n+1} \leq \hat{q}_{1-\alpha})$ è il rango atteso ti s_{n+1} tiviso $n+1$.

Il quantile scelto è al più $\lceil (1-\alpha)(n+1) \rceil / (n+1)$. Quinti:

$$P(y_{n+1} \in C) \leq \frac{\lceil (1-\alpha)(n+1) \rceil}{n+1} \leq \frac{(1-\alpha)(n+1)+1}{n+1} = 1 - \alpha + \frac{1}{n+1}. \quad \square$$

D.3 Soluzioni selezionate dalla Parte C

Esercizio C.5 (DPO: direzione del gradiente). La loss DPO è $\mathcal{L} = -\mathbb{E}[\log \sigma(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\mathrm{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\mathrm{ref}}(y_l|x)})]$.

Sia $\Delta = \beta \log \frac{\pi_\theta(y_w)}{\pi_{\mathrm{ref}}(y_w)} - \beta \log \frac{\pi_\theta(y_l)}{\pi_{\mathrm{ref}}(y_l)}$. Allora $\nabla_\theta \mathcal{L} = -\mathbb{E}[\sigma(-\Delta) \nabla_\theta \Delta]$, tove $\sigma(-\Delta) = 1 - \sigma(\Delta) \geq 0$. Quinti:

$$\nabla_{\log \pi_\theta(y_w)} \mathcal{L} = -\beta \mathbb{E}[\sigma(-\Delta)] < 0 \quad \Rightarrow \text{ aumenta } \log \pi_\theta(y_w), \tag{D.2}$$

$$\nabla_{\log \pi_\theta(y_l)} \mathcal{L} = +\beta \mathbb{E}[\sigma(-\Delta)] > 0 \quad \Rightarrow \text{ timinuisce } \log \pi_\theta(y_l). \tag{D.3}$$

Il fattore $\sigma(-\Delta)$ è grante (prossimo a 1) quanto la policy corrente assegna erroneamente più probabilità a y_l che a y_w: l'aggiornamento è più forte per le coppie più tifficili, fornento un curriculum implicito. □

Esercizio C.7 (Gradient Checkpointing). Un motello a L layer richiete turante il backwart ti accetere alle attivazioni forwart ti ogni layer. Senza checkpointing: memoria $O(L)$.

Con checkpointing ogni k layer: si salva l'attivazione ogni k layer (memoria $O(L/k)$ per i checkpoint). Durante il backwart, per ogni segmento ti k layer tra checkpoint atiacenti si ricalcola il forwart (costo $O(k)$ passi aggiuntivi). La memoria aggiuntiva per il backwart ti ogni segmento è $O(k)$.

La memoria totale è $O(L/k + k)$: i tue termini sono bilanciati quanto $L/k = k$, cioè $k = \sqrt{L}$. Con $k = \sqrt{L}$, la memoria tiventa $O(L/\sqrt{L} + \sqrt{L}) = O(\sqrt{L})$.

Il costo computazionale aggiuntivo è un forwart pass per ogni segmento ti $k = \sqrt{L}$ layer, per $L/k = \sqrt{L}$ segmenti: costo totale aggiuntivo $O(L)$, pari al costo tel forwart originale. Quinti Gratient Checkpointing timezza il costo computazionale aggiuntivo al costo ti riturre la memoria ta $O(L)$ a $O(\sqrt{L})$. □

Appendice E

Glossario Aggiuntivo

Simbolo	Significato
$E_\theta(x)$	Energia di un EBM
$Z(\theta)$	Partition function $\int e^{-E_\theta}\, dx$
$s_\theta(x,t)$	Score network $\approx \nabla_x \log p_t(x)$
$\bar{\alpha}_t$	Prodotto cumulativo $\prod_{s=1}^{t}(1-\beta_s)$ (VP-SDE)
$\mathcal{H}_k$	RKHS associato al nucleo k
ECE	Expected Calibration Error
$\bar{A}, \bar{B}$	Matrici SSM discretizzate (ZOH)
r	Rango LoRA
Δ	Step di discretizzazione SSM
τ	Temperatura (CFG, CLIP, DPO)
Φ^{-1}	CDF inversa normale (in NF4)
$\mathrm{lse}(\beta, z)$	Log-sum-exp $\frac{1}{\beta}\log\sum_i e^{\beta z_i}$
ESS	Effective Sample Size MCMC
$\hat{q}_{1-\alpha}$	Quantile conformal per coverage $1-\alpha$
μ_P	Mean embedding di P nell'RKHS
h_t, s_t	Stato deterministico e stocastico RSSM
N_d	Numero GPU in data parallelism
π_{ref}	Reference policy in RLHF/DPO
C^*	Soglia di compute per emergent abilities
NFE	Number of Function Evaluations (diffusion)
PPL	Perceptual Path Length (StyleGAN)
CB	Codebook (VQ-VAE)
STE	Straight-Through Estimator (VQ-VAE)
w	Guidance weight (CFG)

Appendice F

Tabelle di Riferimento Rapido

F.1 Algoritmi di RL Profondo: Confronto

Algoritmo	Policy	Off-policy	Esplorazione	Modello	Punto di forza
DDPG	Det.	Sì	Rumore OU	No	Semplicità
TD3	Det.	Sì	Rumore clip	No	Stabilità
SAC	Stocast.	Sì	MaxEnt	No	Robustezza
MBPO	Stocast.	Sì	MaxEnt	Sì (ensemble)	Efficienza camp.
Dreamer	Stocast.	Sì	MaxEnt	Sì (RSSM)	Visione/planning

F.2 Modelli Generativi: Confronto Sintetico

Modello	Velocità camp.	Qualità	Diversità	Likelihood
GAN	⋆⋆⋆⋆⋆	⋆⋆⋆⋆	⋆⋆⋆	No
VAE	⋆⋆⋆⋆⋆	⋆⋆⋆	⋆⋆⋆⋆	Approx.
NF	⋆⋆⋆⋆	⋆⋆⋆	⋆⋆⋆⋆	Esatta
DDPM ($T = 1000$)	⋆	⋆⋆⋆⋆⋆	⋆⋆⋆⋆⋆	Approx.
DDIM ($S = 50$)	⋆⋆⋆	⋆⋆⋆⋆⋆	⋆⋆⋆⋆	Approx.
FM ($S = 20$)	⋆⋆⋆⋆	⋆⋆⋆⋆⋆	⋆⋆⋆⋆	Approx.
CM ($S = 1$–4)	⋆⋆⋆⋆⋆	⋆⋆⋆⋆	⋆⋆⋆⋆	Approx.
LDM/SD	⋆⋆⋆⋆	⋆⋆⋆⋆⋆	⋆⋆⋆⋆	Approx.

F.3 Metodi di Inferenza: Confronto

Metodo	Bias	Scalabilità	Garanzie	Uso consigliato
MCMC (HMC)	0	Medio	Asint.	Dataset piccoli, $d < 10^4$
SGLD	Piccolo	Alta	Asint. (lr$\to 0$)	Dataset grandi
CAVI	Sì	Alta	ELBO bound	Modelli strutturati
BBVI	Piccolo	Alta	ELBO bound	Reti neurali
Laplace	Sì (locale)	Alta	Nessuna	Post-hoc su MAP
Ensembles	Nessuno	Bassa ($M\times$)	Empiriche	Alta affidabilità
Conformal	0	Alta	Esatte	Qualsiasi modello

F.4 PEFT: Confronto dei Metodi

Metodo	**Param. tren.**	**Lat. inf.**	**Merge pesi**	**Flessibilità**
Full fine-tuning	100%	1×	N/A	Massima
Adapter ($r = 64$)	$\sim 3\%$	1.03×	No	Media
Prefix ($L_P = 10$)	$\sim 0.1\%$	1×	No	Bassa
LoRA ($r = 8$)	$\sim 0.5\%$	1×	Sì	Media
QLoRA (NF4+LoRA)	$\sim 0.5\%$	1×	Sì	Media

F.5 SSM vs. Transformer: Profilo Computazionale

Modello	**Training**	**Inferenza (per token)**	**Memoria**	**Recall esatto**
Transformer	$O(L^2 d)$	$O(Ld)$	$O(L^2)$	⋆⋆⋆⋆⋆
Longformer	$O(Lwd)$	$O(wd)$	$O(Lw)$	⋆⋆⋆
S4	$O(L \log L \cdot d)$	$O(d_s)$	$O(L)$	⋆⋆
Mamba	$O(Ld)$	$O(d_s)$	$O(L)$	⋆⋆⋆
RWKV	$O(Ld)$	$O(d_h)$	$O(L)$	⋆⋆

d_s = dimensione dello stato SSM ($O(1)$ in numero di token); d_h = dimensione nascosta della RNN.

Note Bibliografiche Estese

Parte A · Modelli Generativi Avanzati e RL Profondo

Haarnoja, T. et al. (2018, 2019). Soft Actor-Critic e temperatura adattiva.

Fujimoto, S. et al. (2018). TD3: Twin Delayed Deep Deterministic Policy Gradient.

Janner, M. et al. (2019). MBPO: Model-Based Policy Optimization.

Aghajanyan, A. et al. (2021). Intrinsic dimensionality of NLP fine-tuning tasks.

Hafner, D. et al. (2019, 2020, 2023). RSSM, Dreamer, DreamerV3: world models per planning visivo.

Williams, G. et al. (2017). MPPI: Model Predictive Path Integral.

Song, Y. & Ermon, S. (2019). Score matching con dinamica di Langevin.

Song, Y. et al. (2020, 2021). SDE framework unificante per modelli a diffusione.

Ho, J. et al. (2020). DDPM: Denoising Diffusion Probabilistic Models.

Song, J. et al. (2021b). DDIM: Campionamento deterministico per modelli diffusivi.

Lipman, Y. et al. (2022). Flow Matching e Conditional Flow Matching.

Song, Y. et al. (2023). Consistency Models: campionamento in un passo.

Ho, J. & Salimans, T. (2022). Classifier-Free Guidance per modelli diffusivi.

Rombach, R. et al. (2022). Latent Diffusion Models: Stable Diffusion.

Hinton, G. (2002). Contrastive Divergence per Energy-Based Models.

Ramsauer, H. et al. (2021). Modern Hopfield Networks e connessione con l'attenzione.

van den Oord, A. et al. (2017). VQ-VAE: Vector Quantized Variational Autoencoder.

Razavi, A. et al. (2019). VQ-VAE-2: struttura gerarchica e generazione autoregressiva.

Chang, H. et al. (2022). MaskGIT: generazione bidirezionale iterativa.

Heusel, M. et al. (2017). FID: Fréchet Inception Distance.

Salimans, T. et al. (2016). Inception Score per la valutazione dei modelli generativi.

Radford, A. et al. (2021). CLIP: Learning Transferable Visual Models from Natural Language.

Kynkäänniemi, T. et al. (2019). Improved Precision and Recall for generative models.

Karras, T. et al. (2019, 2020). StyleGAN e StyleGAN2: disentanglement dello spazio latente.

Grill, J.-B. et al. (2020). BYOL: Bootstrap Your Own Latent.

Zbontar, J. et al. (2021). Barlow Twins: self-supervised learning senza campioni negativi.

Karras, T. et al. (2023). Analisi della curvatura delle traiettorie FM vs. DDPM.

Parte B · Metodi Kernel, MCMC e Inferenza Variazionale

Mercer, J. (1909). Teorema di Mercer: decomposizione spettrale dei nuclei PSD.

Moore, E.H. (1908). Generalizzazione degli spazi di Hilbert a nucleo riproducente.

Aronszajn, N. (1950). Theory of Reproducing Kernels.

Boser, B. et al. (1992). Support Vector Machine per la classificazione non lineare.

Gretton, A. et al. (2012). Maximum Mean Discrepancy e two-sample test.

Metropolis, N. et al. (1953). Algoritmo Metropolis-Hastings per il campionamento MCMC.

Neal, R. (2011). MCMC using Hamiltonian Dynamics: HMC review.

Hoffman, M. & Gelman, A. (2014). NUTS: No-U-Turn Sampler.

Welling, M. & Teh, Y.W. (2011). SGLD: Stochastic Gradient Langevin Dynamics.

Geyer, C.J. (1991). Parallel Tempering: campionamento di distribuzioni multimodali.

Jordan, M. et al. (1999). CAVI: Coordinate Ascent Variational Inference.

Ranganath, R. et al. (2014). BBVI: Black Box Variational Inference.

Kingma, D. & Wierstra, M. (2014). Auto-encoding variational Bayes e reparametrization trick.

Burda, Y. et al. (2016). IWAE: Importance Weighted Autoencoders.

Lakshminarayanan, B. et al. (2017). Deep Ensembles per la stima dell'incertezza.

Guo, C. et al. (2017). On calibration of modern neural networks: temperature scaling.

Vovk, V. et al. (2005). Conformal Prediction: garanzie di copertura senza assunzioni distributive.

Liu, W. et al. (2020). Energy Score per Out-of-Distribution Detection.

Higgins, I. et al. (2017). beta-VAE: Learning Basic Visual Concepts.

Ferguson, T. (1973). Dirichlet Process: stima bayesiana non parametrica.

Neal, R. (1996). Bayesian Learning for Neural Networks: GP come limite di BNN.

Elhage, N. et al. (2022). Sparse Autoencoders per la mechanistic interpretability.

Parte C · Transformer Efficienti, SSM, Fine-Tuning e Allineamento

Dao, T. et al. (2022, 2023). FlashAttention v1 e v2: IO-aware exact attention.

Milakov, M. & Gimelshein, N. (2018). Online normalizer calculation for softmax.

Katharopoulos, A. et al. (2020). Transformers are RNNs: Linear Attention.

Choromanski, K. et al. (2021). Performer: Random Feature Attention.

Su, J. et al. (2023). RoPE: Rotary Position Embedding.

Press, O. et al. (2022). ALiBi: Train Short, Test Long.

Gu, A. et al. (2020). HiPPO: Recurrent Memory with Optimal Polynomial Projections.

Gu, A. et al. (2022a). S4: Efficiently Modeling Long Sequences with Structured State Spaces.

Gu, A. et al. (2022b). S4D: On the Parameterization and Initialization of Diagonal State Spaces.

Gu, A. & Dao, T. (2023). Mamba: Linear-Time Sequence Modeling with Selective State Spaces.

Peng, B. et al. (2023a). RWKV: Reinventing RNNs for the Transformer Era.

Peng, B. et al. (2023b). YaRN: Efficient Context Window Extension of LLMs.

Hu, E. et al. (2022). LoRA: Low-Rank Adaptation of Large Language Models.

Dettmers, T. et al. (2023). QLoRA: Efficient Finetuning of Quantized LLMs.

Houlsby, N. et al. (2019). Parameter-Efficient Transfer Learning with Adapter modules.

Li, X. & Liang, P. (2021). Prefix-Tuning: Optimizing Continuous Prompts for Generation.

Rajbhandari, S. et al. (2020). ZeRO: Memory Optimizations Toward Training Trillion Parameter Models.

Zhao, Y. et al. (2023). FSDP: Fully Sharded Data Parallel.

Chen, T. et al. (2016). Training Deep Nets with Sublinear Memory Cost (Gradient Checkpointing).

Micikevicius, P. et al. (2018). Mixed Precision Training con FP16 e BF16.

Ouyang, L. et al. (2022). InstructGPT: Training Language Models to Follow Instructions with RLHF.

Rafailov, R. et al. (2023). DPO: Direct Preference Optimization.

Azar, M. et al. (2024). Analisi critica di DPO: collasso della policy di riferimento.

Meng, Y. et al. (2024). SimPO: Simple Preference Optimization senza reference policy.

Bai, Y. et al. (2022). Constitutional AI: Harmlessness from AI Feedback.

Wei, J. et al. (2022). Emergent Abilities of Large Language Models.

Schaeffer, R. et al. (2023). Are Emergent Abilities of LLMs a Mirage?

Xie, S. et al. (2022). An Explanation of In-Context Learning as Implicit Bayesian Inference.

Wang, X. et al. (2023). Self-Consistency Improves Chain of Thought Reasoning.

Lewis, P. et al. (2020). RAG: Retrieval-Augmented Generation for NLP.

Liu, H. et al. (2023). LLaVA: Visual Instruction Tuning per modelli multimodali.